Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Robert Ćorluka

Prikaz uporabe sistema za upravljanje poslovnih pravil Drools na primeru izračuna plač

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Prikaz uporabe sistema za upravljanje poslovnih pravil Drools na primeru izračuna plač

Tematika naloge:

Sistemi za upravljanje s poslovnimi pravili omogočajo ločevanje zapisa poslovnih pravil od izvorne kode aplikacije, kar poleg ostalih ugodnih učinkov omogoča predvsem učinkovito prilagajanje aplikacije v primeru sprememb pravil. V diplomski nalogi predstavite prednosti uporabe tovrstnih sistemov v primerjavi z običajnim razvojem poslovne aplikacije. V ta namen predstavite in uporabite odprtokodno platformo Drools za izdelavo spletne rešitve za izračun plače redno zaposlenega, ki je tipičen primer tovrstne poslovne aplikacije. Po prikazu izdelave rešitve le-to primerjajte z običajno rešitvijo za izračun plač ter izpostavite njene prednosti in slabosti.

Zahvaljujem se podjetju SRC d.o.o za pomoč pri ideji in ustvarjanju diplomske naloge in viš. pred. dr. Igorju Rožancu za nasvete in usmeritve glede pisanja diplomske naloge. Posebno zahvalo namenjam Tjaši, ki me je podpirala in mi stala ob strani.

Diplomsko nalogo posvečam vsem, ki si želijo spoznati Droolse in poslovna pravila.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application programming interface	aplikacijski programski vmesnik
BLIP	Business Logic Integration Platform	platforma za integracijo poslovne logike
BRA	Business Rules Approach	pristop s poslovnimi pravili
BRG	Business Rules Group	organizacija oz. skupina, ki se ukvarja s pristopom, ki uporablja poslovna pravila
BRMS	Business Rule Management System	sistem za upravljanje s poslovnimi pravili
DRL	Drools Rule Language	programski jezik za pisanje poslovnih pravil
HTML	Hyper Text Markup Language	označevalni jezik za izdelavo spletnih strani
KIE	Knowledge Is Everything	naziv skupine, ki se ukvarja z odprtokodnimi rešitvami za poslovne sisteme
ODM	Operations Decision Manager	celostni BRMS sistem podjetja IBM
PHP	Hypertext Preprocessor	skriptni programski jezik za razvoj dinamičnih spletnih strani
REST	Representational State Transfer	arhitektura za izmenjavo podatkov med spletnimi storitvami

Povzetek

Naslov: Prikaz uporabe sistema za upravljanje poslovnih pravil Drools na primeru izračuna plač

V diplomski nalogi je opisana uporaba sistemov za upravljanje s poslovnimi pravili. Sistem za upravljanje s poslovnimi pravili je celosten sistem, pri katerem je glavni cilj, da je odločitvena logika ločena od izvorne kode aplikacije. V okviru tega dela so predstavljena tudi glavna načela tovrstnih sistemov, nekateri najbolj znani sistemi ter njihove prednosti in slabosti.

Prikazan je tudi primer uporabe enega izmed sistemov za upravljanje s poslovnimi pravili in sicer odprtokodna platforma Drools. Primer uporabe te platforme je prikazan na primeru izračuna plač za redno zaposlene. V diplomski nalogi spoznamo tudi jezik za pisanje poslovnih pravil Drools Rule Language, katerega tudi uporabimo na primeru izračuna plače.

Na koncu lahko razberemo, da platforma Drools omogoči lažje in hitrejše prilagajanje ter vzdrževanje poslovnih pravil in razbremeni razvijalce v podjetju. Na podlagi primera za izračun plač izvedemo primerjavo s primerljivo spletno aplikacijo, ki ne uporablja platforme Drools. Ugotovimo lahko, da prihranimo 71% vrstic izvorne kode, hkrati pa omogočimo preproste in hitre prilagoditve. Po anketi s razvijalci in poslovnimi analitiki ugotovimo, da bi jim uporaba platforme Drools olajšala delo v primeru, ko projekti zahtevajo veliko sprememb poslovnih pravil v kratkem časovnem obdobju.

Ključne besede: Sistem za upravljanje s poslovnimi pravili, BRMS, Drools, analiza, izračun plač.

Abstract

Title: The demonstration of the BRMS Drools use shown on the salary calculation example

The thesis describes the use of business rule management system. A business rule management system (BRMS) is an integrated system which has the primary goal to seperate the decision logic from the source code of the application. We will present BRMS's main principles, the best-known BRMS systems and their advantages and disadvantages.

The demonstration is presented by the use of open source BRMS system Drools. An example is shown for the salary calculation example for regular employees. We also learn about the buissnes rule language for writing rules called Drools Rule Language (DRL). Finally, we prove the Drools system makes it easier and faster to customize and maintain business rules, and saves time to the developers in the company.

Based on the salary calculation example the comparison with a similar non-BRMS web application is done. We found out that we can save 71% lines of the source code while allowing simple and quick adjustments of business rules. A survey with developers and business analysts proved using Drools makes it easier to work with projects that require a lot of changes of business rules over a period of time.

Keywords: Business Rule Management System, BRMS, Drools, analysis, salary calculation example.

Poglavje 1

Uvod

V današnjem času se podjetja soočajo s velikimi pritiski. Vsakodneven pojav nove konkurence, hitro spreminjajoča se nova zakonodaja, vse višji prodajni cilji, višje zahteve strank se izražajo tudi v stalnem spreminjanju poslovnih pravil. Problem nastane predvsem tedaj, ko se pravila začnejo prepletati ali zaradi slabe definicije in niso ločena od implementacije aplikacije. To pomeni, da jih lahko spreminja samo programer, ki jih je tudi napisal.

Sistem za upravljanje s poslovnimi pravili (angl. Business Rule Management System - BRMS) omogoča ločeno pisanje in preverjanje pravil, katera lahko napiše tudi oseba brez programerskega predznanja [27]. V tem primeru aplikacija dostopa do pravil preko BRMS sistema in jih potemtakem ne vsebuje v sebi, s tem pa omogoča poslovnemu analitiku hitro in učinkovito spremembo pravil, ki prilagodi delovanje aplikacije spremembam okoliščin (npr. novi zakonodaji pri izračunu plač, novi pogojem za pridobitev kredita, itd.)

Cilj diplomske naloge je spoznati nekaj več o sistemih za upravljanje s poslovnimi pravili, predvsem spoznati enega tovrstnih sistemov Drools, ki je izdelek podjetja Red Hat, ter način tvorbe poslovnih pravil v programskem jeziku Drools Rule Language [12, 11]. V drugem delu bo prikazan praktičen primer spletne aplikacije za izračun plač, ki uporablja poslovna pravila napisana s pomočjo jezika Drools Rule Language. Predstavljen bo tudi del platforme Drools za izvajanje poslovnih pravil, t.j. KIE izvedbeni strežnik

(angl. KIE Execution server) [13]. Naloga bo zaključena s primerjalno analizo tovrstne izvedbe s klasično rešitvijo, ki vsebuje v izvorno kodo vgrajena pravila.

Poglavje 2

Ozadje

2.1 Sistem za upravljanje s poslovnimi pravili

Sistem za upravljanje s poslovnimi pravili (angl. Business Rule Management System - BRMS) je celosten sistem, kjer se odločitvena logika uporabi za izvedbo poslovnih pravil, ki so nato uporabljna pri odločanju v različnih delih aplikacij. Sistem omogoča definiranje, shranjevanje, izvajanje, urejanje, opazovanje in vzdrževanje poslovnih pravil. Namesto vdelanih odločitev v aplikacijo se v BRMS sistemu pravila izloči in se upravlja z njimi zunaj aplikacije. S tem se omogoči uporabo istih pravil za več aplikacij in lažje vzdrževanje le-teh, saj je potrebno spremeniti pravila le takrat, ko ta vplivajo na delovanje vseh aplikacij, ki jih uporabljajo [27, 33].

2.2 Pristop s poslovnimi pravili

Pristop s poslovnimi pravili (angl. Business Rules Approach - BRA) je razvojna metadologija, kjer so pravila sicer uporabljena, vendar niso neposredno implementirana v programski kodi. Ta pristop formalizira kritična komercialna poslovna pravila v jeziku, ki ga lahko razumejo poleg razvojnikov tudi uporabniki brez znanja programiranja [4, 6].

2.2.1 Načela neodvisnosti pravil

Ključna načela uporabe pristopa s poslovnimi pravili so predstavljena v t.i. Business Rules Manifesto, ki je napisan s strani skupine Buisness Rules Group. Ta se ukvarja s tem pristopom že od leta 1989 [5, 6].

Primarne zahteve, ki niso sekundarne

Pravila so osnova za poslovne in tehnološke modele in so zelo pomembni pri zahtevah.

Ločeno od procesov, ki jih ne vsebujejo

Pravila veljajo za vse procese in procedure. Obstajati mora enoten sklop pravil, ki se dosledno izvaja na vseh pomembnih področjih poslovne dejavnosti. Pravila niso ne proces in ne procedura, zato ne smejo biti vsebovane niti v procesih niti v procedurah. Pravila so tudi izrecne omejitve vedenja, ki omogočajo podporo vedenju.

Namerno znanje, ne stranski proizvod

Pravila temeljijo na dejstvih, dejstva pa temeljijo na konceptih, izraženih z pogoji. Pogoji izražajo poslovne koncepte, dejstva dajejo trditve o teh konceptih, pravila omejujejo in podpirajo ta dejstva. Pravila so temelj za to, kar podjetje ve o sebi, to je osnovno poslovno znanje. Pravila je treba vzdrževati, varovati in upravljati.

Deklarativno, ne postopkovno

Pravila naj bodo deklarativno izražena v stavkih v naravnem jeziku, saj so namenjena poslovnemu občinstvu. Pravilo se razlikuje od vseh izvajanj, ki so za to pravilo določena. Pravilo in njegovo izvajanje sta ločeni zadevi. Če se nekaj ne da izraziti, potem to ni pravilo. Izjeme za pravila so izražene z drugimi pravili.

Dobro oblikovan izraz in ne ad hoc

Poslovna pravila morajo biti izražena tako, da jih lahko potrdijo poslovni ljudje. Tudi izražena morajo biti v smislu, da jih je mogoče preveriti med seboj zaradi skladnosti. Formalna logika (recimo logika predikatov), je bistvena za dobro oblikovano izražanje pravil v poslovnem smislu, pa tudi za tehnologije, ki implementirajo poslovna pravila.

Arhitektura, ki temelji na pravilih, in ne posredna implementacija

Aplikacija, ki uporablja poslovna pravila, je namenoma prilagojena nenehnemu spreminjanju poslovnih pravil. Platforma, na kateri teče aplikacija, bi morala podpirati takšne stalne spremembe. Razmerje med dogodki in pravili je ponavadi mnogo proti mnogo.

Po pravilih vodeni procesi in ne programiranje na osnovi izjeme

Pravila določajo mejo med sprejemljivo in nesprejemljivo poslovno dejavnostjo. Pravila pogosto zahtevajo posebno ali selektivno obravnavanje ugotovljenih kršitev. Kršitev pravil je dejavnost kot vsaka druga dejavnost. Da bi zagotovili maksimalno doslednost in ponovno uporabnost, bi bilo treba ravnanje z nesprejemljivo poslovno dejavnostjo ločiti od ravnanja sprejemljive poslovne dejavnosti.

Za dobro podjetja, ne za tehnologijo

Pravila se nanašajo na poslovno prakso in usmerjanje, zato pravila temeljijo na poslovnih ciljih in jih oblikujejo različni vplivi. Pravila vedno nekaj stanejo podjetje. Stroški uveljavljanja poslovnih pravil morajo biti uravnoteženi glede na poslovna tveganja in poslovne priložnosti, ki bi se sicer lahko izgubile. Več pravil ni boljše. Običajno je manj dobrih pravil boljše.

Namenjeno bolj poslovnim ljudem kot pa osebam na področju informacijskih tehnologij

Pravila bi morala izhajati iz izkušenih poslovnih ljudi. Poslovni ljudje morajo imeti na voljo orodja, ki jim pomagajo oblikovati, preverjati in upravljati pravila.

Upravljanje poslovne logike in ne strojne / programske opreme platforme

Poslovna pravila so bistveno poslovno sredstvo. Dolgoročno so pravila za podjetja pomembnejša od strojne in programske platforme. Pravila morajo biti organizirana in shranjena tako, da jih je mogoče zlahka namestiti na nove platforme strojne in programske opreme. Pravila in njihova zmožnost učinkovitega spreminjanja so bistvenega pomena za izboljšanje poslovne prilagodljivosti.

2.3 Najbolj znani BRMS sistemi

Obstaja veliko različnih BRMS sistemov, večina jih temelji na programskem jeziku Java. Poseben je sistem InRule BRMS [18], saj omogoča podporo programskemu jeziku .NET.

2.3.1 Najboljši samostojni BRMS sistemi

V tej kategoriji so celoviti sistemi, ki imajo razširjen nabor možnosti in dobro narejene komponente, ki lahko podpirajo več podjetij in hkrati zmanjšajo stroške vzdrževanja s strani programerjev.

- **IBM Operations Decision Manager (ODM)** [17]
  - Ima bogat nabor možnosti, ampak visoko ceno. Primeren za podjetja, ki opravljajo poslovno analitiko in dogodke zahtevnih procesov.

- **FICO Blaze Advisor Decision Rules Management System** [15]
  - Ima bogat nabor možnosti. Cena je srednje visoka. Uporabljajo ga predvsem podjetja, ki se ukvarjajo s financami.
- **Progress Corticon** [25]
  - Je dobro narejen in enostaven za uporabo. Cenejši od konkurenčnih IBM ODM in od FICO Blaze.

2.3.2 Najboljši odprtokodni BRMS sistemi in orodja

Odprtokodni BRMS sistemi oziroma orodja za poganjanje pravil so bolj osredotočeni za razvijalce. Ko je ogrodje zgrajeno, ta odprtokodna orodja za pravila služijo svojemu namenu, vendar jim običajno manjkajo napredne opcije, ki jih ponujajo prej omenjeni samostojni BRMS sistemi. Lahko pa jih programerji potem sami pripravijo, vendar je potem to posebej prirejeno potrebam podjetja. Tako, da so odprtokodni sistemi bolj usmerjeni podjetjem, ki si želijo ustvariti BRMS sistem po svojih željah.

- **Drools** [32]
  - Ponuja dosti izbire orodij kot so Drools Expert, Drools Fusion in jBPM [10]. Ima zelo dobro napisano dokumentacijo. Platforma omogoča tudi implementacijo in poganjanje bolj zahtevnih pravil. Še vedno se izboljšujejo in ponujajo redne popravke napak, ki so odkrite s strani uporabnikov.
- **OpenRules** [22]
  - Je preprost za uporabo. Omogoča med drugim uporabnikom tudi uporabo pravil v Google dokumentih ali pa v Excel datotekah. Primeren je za manjša podjetja oziroma manjše aplikacije, ki jih uporabljajo ta podjetja.
- **JESS** [19]

– Ponuja samo orodje za izvedbo poslovnih pravil (angl. rule engine). Primeren le za manjše poslovne aplikacije. Ker gre za starejši sistem, ne ponuja veliko podpore in se ne posodablja ter nadgrajuje.

V okviru izdelave te diplomske naloge smo se odločili za odprtokodno platformo Drools, katero bomo podrobneje predstavili v nadaljevanju. Izbrali smo jo, ker je zelo prilagodljiva in ima dobro napisano dokumentacijo.

2.4 Platforma Drools

Drools je platforma za integracijo poslovne logike (angl. Business Logic Integration Platform - BLIP) [32]. Je odprtokodni projekt, napisan v programskem jeziku Java in podprt s strani podjetja Red Hat. Platforma Drools je dejansko zbirka več orodij, ki nam omogočajo ločitev logike in podatkov znotraj poslovnih procesov. Razdeljen je na dva ključna dela, ki sta zapisovalni del (angl. authoring) in izvajalni del (angl. runtime). Zapisovalni del je namenjen ustvarjanju datotek s pravili (`.drl` končnica) Izvajalni del pa omogoča ustvarjanje delovnega spomina in poganjanje pravil [32, 13, 30]. V nadaljevanju bom predstavil orodji Drools Expert in Drools Workbench ter predstavil programski jezik Drools Rule Language.

2.4.1 Orodje Drools Expert

Orodje Drools Expert (poznan tudi po starem imenu KIE Execution Server) je orodje za izvrševanje poslovnih pravil (angl. rule engine), torej sistem, ki uporablja pristop s pravili za izvrševanje določenih primerov. Uporablja deklarativni način, ki nam omogoča , da definiramo, kaj storiti in ne kako to narediti [14].

Prednosti

Predstavili bomo nekaj ključnih prednosti uporabe orodja Drools Expert, ki so sledeča:

- Integracija z orodji
  - Orodja za razvijanje (npr. Eclipse) ponujajo veliko načinov za urejanje in upravljanje s pravili. Omogočajo nam tudi razhroščevanje (angl. debugging) pravil.
- Deklarativno programiranje
  - Pravila nam olajšajo delo, saj lahko izrazimo težke probleme na preprost način. Pravila so napisana v manj zahtevnem jeziku, zato jih lahko urejajo ljudje brez računalniškega predznanja (npr. poslovni analitiki).
- Hitrost in skalabilnost
  - Algoritem Rete OO [26], ki ga platforma Drools uporablja, je že dobro uveljavljen algoritem. S pomočjo platforme Drools lahko naša aplikacija postane zelo skalabilna.
- Ločevanje logike od podatkov
  - Poslovna logika je v sistemu Drools ločena od podatkov. Odvisno od projekta, vendar je v vsakem primeru ločitev lahko zelo koristna pri projektu.

2.4.2 Drools Rule Language

Vsaka datoteka s pravili (`.drl` končnica) mora vsebovati sledeče stvari:

- Paket (angl. package)

  - Vsaka datoteka s pravili se začne s imenom paketa (angl. package name). Paket deluje kot imenski prostor (angl. namespace) za pravila. Imena pravil znotraj paketa morajo biti edinstvena (angl. unique). Paketi v pravilih so zelo podobni paketom v programskem jeziku Java. Primer lahko vidimo na sliki 2.1 v prvi vrstici.

- Uvozi (angl. imports)

  - Uvozi so tudi enaki kot v programskem jeziku Java. Uporabljajo se za določene objekte, katere želimo uporabiti v naših pravilih. Primer za objekt Student vidimo na sliki 2.1

primer_pravil.drl

```
1  package si.primer.pravil;
2
3  import si.primer.Student;
4
5  rule primerpravilal
6  when
7      Student( ime == "Robert" )
8  then
9      System.out.println("Sporzi se, ko dobi objekt Student z atributom ime Robert");
10 end
11
12 rule vedno
13 when
14 then
15     System.out.println("To pravilo se sprozi v vsakem primeru !");
16 end
```

Slika 2.1: Primer datoteke s pravili

- Definicija pravil

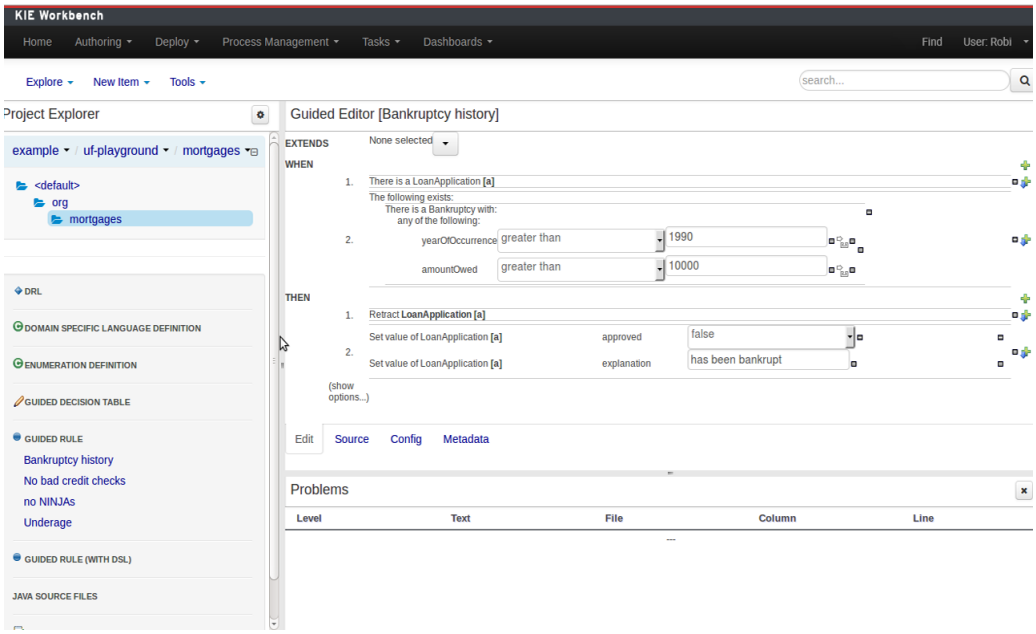
  - Pravilo je sestavljeno iz dveh delov in sicer iz pogoja (angl. when) in posledice (angl. then) dela. Ko je pogojni del izpolnjen se izvrši akcija v posledičnem delu [11, 12]. Ključne besede so `"rule","when", "then"` in `"end"`, kot vidimo spodaj na primeru:
    Primer:

```
rule ime_pravila
when
```

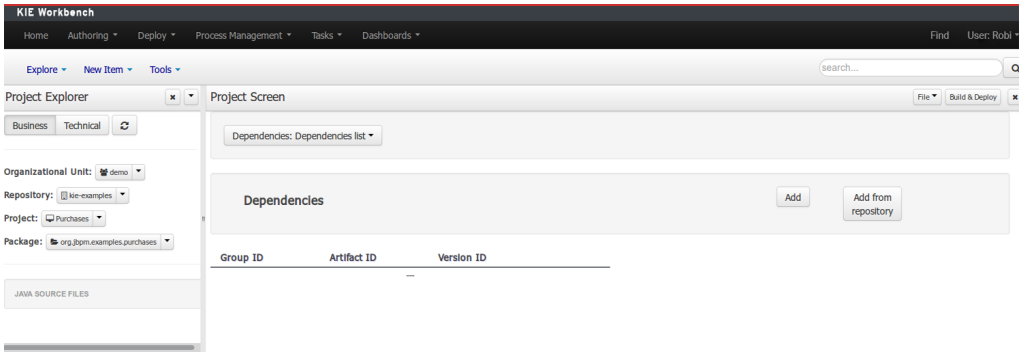
```
            Student( ime == "Robert" )
    then
            System.out.println("Testno pravilo");
    end
```

2.4.3 Drools Workbench

Drools Workbench (poznan tudi po starem imenu KIE Workbench) je spletno okolje, kjer lahko ustvarjamo preko spletnega vmesnika pravila, procese in podatkovne modele. Urejanje projekta vidimo na sliki 2.3 [13]. Omogoča, da uporabniki brez računalniškega predznanja preprosto vnesejo pravila, primer vidimo na sliki 2.2.



Slika 2.2: Urejanje pravil



Slika 2.3: Urejanje projekta

2.5 Prednosti uporabe BRMS sistema

Predstavili bomo nekaj ključnih prednosti uporabe BRMS sistema [3].

- Zmanjšana odvisnost od razvijalcev
  - Ta prednost je ključna, saj je celoten postopek priprave pravil pravzaprav namenjen ljudem brez programerskega znanja kot so poslovni analitiki. To je velika prednost za podjetje, saj je časovna zamuda med vpeljavo novega pravila v tradicionalnem programskem okolju in BRMS dovolj velika, da upravičuje uporabo sistema BRMS.
- Testiranje in zagotavljanje kakovosti
  - Testi morajo biti prisotni, da se zagotovi celovitost sistema za vsako posodobitvijo pravil. Upravljanje pravil lahko kontrolirajo tudi poslovni analitiki ali pa kdo drug, da se ne obremenjuje razvijalcev. To pomeni, da tudi v primeru sprememb ne more priti do nesporazumov med analitiki in razvijalci, saj so tu razvijalci izvzeti in so le analitiki tisti, ki skrbijo za pravila.
- Visoka povezanost s poslovnim jezikom

  - Ker obstaja visoka povezanost med poslovnim jezikom in razvojem pravil, se lahko pravila zapišejo tako, da jih lahko prebere vsakdo, ki razume načela poslovanja. V prednosti je bistveno, da za to ne potrebuje tehničnega predznanja.

- Povezava s drugimi sistemi

  - Avtomatizirane procese je mogoče uporabiti, da uveljavljajo nova pravila. To lahko podjetjem zelo pomaga pri prilagajanju pravil glede na stanje na trgu in pred konkurenco.

2.6 Slabosti uporabe BRMS sistema

Predstavili bomo tudi nekaj ovir pri uporabi BRMS sistemov [3].

- Postavitev sistema

  - Za postavitev in razvoj svojega BRMS sistema so potrebni strokovnjaki s poglobljenim znanjem o BRMS sistemih, poslovnih pravilih, razvoju poslovnih aplikacij in dobre komunikacijske sposobnosti, da uspešno razumejo poslovne prakse in znanja iz poslovnega sveta. Takih strokovnjakov je zelo malo in njihova zaposlitev lahko podjetje drago stane.

- Odvisnost od razvijalcev

  - Kljub cilju BRMS sistemov, da se vnos pravil čim bolj oddalji od razvijalcev, so le-te v realnosti še vedno potrebni, saj se pri zahtevnejših pravilih in pri oblikovanju posebnih podatkovnih objektov in struktur znanje poslovnega analitika ne zadošča.

- Navezanost na določenega proizvajalca

  - V primeru uporabe BRMS sistema enega proizvajalca postane podjetje odvisno od tega. Zaradi tega v primeru, če se podjetje

odloči zamenjati BRMS sistem za sistem katerega drugega ponudnika, jih to lahko drago stane in sicer časovno in denarno.

- Zmogljivost BRMS sistema
  - Sistemi BRMS niso optimizirani za implementacijo zahtevnih algoritmov in izračunov, saj zmogljivost izvajanja pravil, ki upravlja s zahtevnimi algoritmi in izračuni drastično upade in posledično lahko pride tudi do napake pri izvajanju le-teh.

Poglavje 3

Izbran tipičen primer uporabe poslovnih pravil

V okviru tega poglavja bo predstavljena ideja tipičnega primera za uporabo poslovnih pravil. Predstavili bomo tudi katere tehnologije in na kakšen način smo jih pripravili za uporabo. Predstavili bomo tudi del platforme Drools, ki smo ga uporabili, uporabljene vsebnike Docker, spletne tehnologije za spletni vmesnik in nazadnje še pojasnili uporabo programa Microsoft Excel za oblikovanje pravil.

3.1 Ideja primera

Dandanes je aktualno, da si preverimo izračun plače preko spletne aplikacije za izračun plače. Zato bomo prikaz uporabe platforme Drools prikazali na praktičnem primeru izračuna plač s pomočjo spletne aplikacije. Platformo Drools smo izbrali predvsem zato, ker je odprtokodna platforma, torej je na voljo brezplačno in omogoča prilagoditve ter uporabo samo določenih delov platforme.

Preko naše spletne aplikacije si lahko izračunamo neto plačo iz bruto plače in obratno, izračunamo pa tudi strošek podjetja za podano neto ali bruto plačo. Izračun upošteva tudi dodatne olajšave kot so: vzdrževani otroci,

bonitete, starost in invalidnost.

Primer izračuna plač je primeren za prikaz uporabe platforme Drools, ker se pogosto dogaja, da se spreminja zakon glede raznih olajšav in davkov, kar seveda vpliva na izračun. Primer izračuna plač je tudi dovolj zahteven, saj je potrebno upoštevati različne olajšave kot so bonitete, vzdrževane otroke in invalidnost. Za vsako od omenjenih olajšav bomo napisali svoje poslovno pravilo, ki bo v skladu s zakonom in stopnjami veljavnih za tekoče leto [24, 8, 7].

V nadaljevanju bomo predstavili uporabljene elemente pri pripravi naše spletne aplikacije za izračun plače redno zaposlenega.

3.2 Platforma Drools

Od platforme Drools smo uporabili KIE izvedbeni strežnik. To je samostojen (angl. standalone) strežnik, ki lahko izvaja pravila. Ta strežnik se pridobi v obliki arhiva spletne aplikacije (angl. web application archive - WAR) [20]. Za poganjanje take vrste aplikacijskega arhiva potrebujemo še strežnik WildFly [28].

Zaradi lažje namestitve bomo uporabil že predpripravljen Docker vsebnik, katerega bomo predstavili v poglavju 3.3. Uporabili smo tudi programski jezik Drools Rule Language (DRL), katerega smo uporabil za pripravo pravil za izračun plače. Zaradi lažje preglednosti smo ta jezik uporabili v programu Microsoft Excel, kateri bo podrobneje opisan v poglavju 3.4.

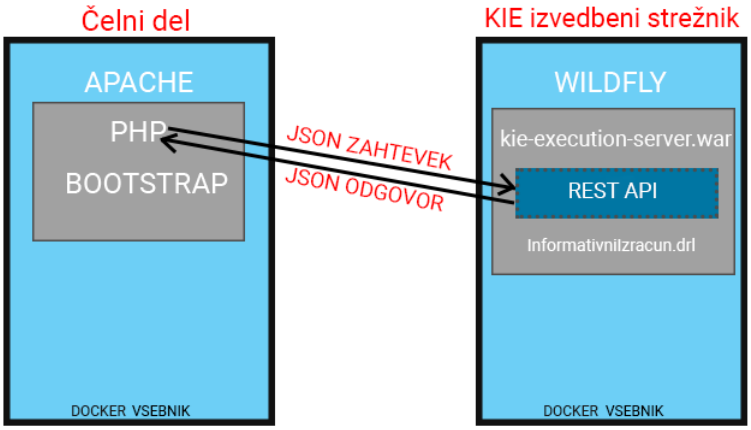
3.3 Vsebnik Docker

Program Docker omogoča virtualizacijo vsebnikov z različno vsebino. Namenjen je postavljanju vsebnikov (angl. container), ki lahko vsebujejo orodja, aplikacije, knjižnice itd. Priročen je za pakiranje aplikacij v vsebnik, ki ga lahko prenašamo in namestimo tudi drugje.

Pri izdelavi smo ločili čelni del oziroma spletni vmesnik in KIE izvedbeni

strežnik in umestili vsakega v svoj Docker vsebnik [9, 31]. Za spletni vmesnik smo uporabili Docker vsebnik s prednaloženim Apache strežnikom. Ta je nastavljen tako, da se naše pripravljene datoteke s spletno aplikacijo naložijo nanj [1]. Za vsebnik s KIE izvedbenim strežnikom, smo uporabili vsebnik s prednaloženim WildFly strežnikom, katerega smo nastavili tako, da naloži WAR datoteko s KIE izvedbenim strežnikom [29].

Oba Docker vsebnika imata vsak svojo nastavitveno datoteko (angl. dockerfile), ki je namenjena za poganjanje Docker vsebnikov. Na sliki 3.1 lahko vidimo shematski prikaz povezav med čelnim delom in strežnikom. Na sliki vidimo na levi strani predstavljen Docker vsebnik z Apache strežnikom, kateri ima napisano spletno aplikacijo v programskem jezku PHP. Iz spletne aplikacije se pošlje JSON zahtevek s podatki, ki jih vnesemo preko vnosnih polj. Na desni strani vidimo drugi Docker vsebnik s WildFly strežnikom, ki pa poganja KIE izvedbeni strežnik. Ta pa ima odprt aplikacijski programski vmesnik (angl. Application Programming Interface - API), na katerega lahko pošljemo JSON zahtevek s podatki in dobimo JSON odgovor s rezultatom.



Slika 3.1: Shematski prikaz docker vsebnikov

Izgled Dockerfile datoteke lahko vidimo na sliki 3.2. Na sliki lahko vidimo, da ko se postavi Docker vsebnik, se še dodatno v virtualnem okolju prenese datoteke iz direktorija `customization` v direktorij `public_html` na virtualnem okolju v vsebniku. Nato sledi še ukaz za posodobitev paketov ter

ukaz za namestitev paketa PHP. Ostalo so ukazi v komentarjih za pomoč pri postavitvi teh Docker vsebnikov.

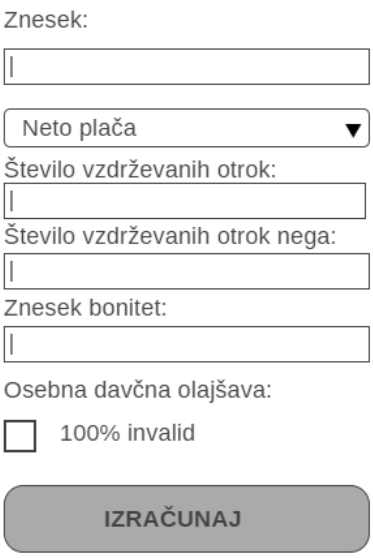
Dockerfile

```
#     #!/bin/bash
#     # Delete all containers
#     docker rm $(docker ps -a -q)
#     # Delete all images
#     docker rmi $(docker images -q)

#docker build -t src-frontend-infoizracun-container .
#run

FROM linode/lamp

MAINTAINER Robert Corluka <robert.corluka@src.si>
USER root
#ADD opla-rules-kie/ /opt/jboss/wildfly/workspace/opla-rules-kie/

###### INSTALL PACKAGES ######
#RUN yum install -y net-tools
#RUN yum install -y wget
#RUN wget http://mirror.cc.columbia.edu/pub/software/apache/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.tar.gz
#RUN tar xzf apache-maven-3.5.0-bin.tar.gz -C /usr/local
#RUN cd /usr/local
#RUN ln -s /usr/local/apache-maven-3.5.0 /usr/local/maven
#RUN export=/usr/local/maven/bin:$PATH


#Customizing wildfly server
ADD customization/ /var/www/example.com/public_html/
RUN apt-get update
RUN apt-get install php5-curl -y
CMD apachectl -D FOREGROUND

#Start docker kie-server
#docker run --privileged=true --net javaapp -p 4002:80 --name src-frontend-infoizracun -i -t src-frontend-infoizracun-container


###sudo docker run --privileged=true --net javaapp -p 4002:80 --name src-frontend-infoizracun -i -t src-frontend-infoizracun-container
###sudo docker build -t src-frontend-infoizracun-container .
###
###go to url http://localhost:4002/demo-informativni-izracun/
#docker rm $(docker ps -a -q -f name="src-frontend-infoizracun")

```

Slika 3.2: Primer Dockerfile nastavitev za čelni del

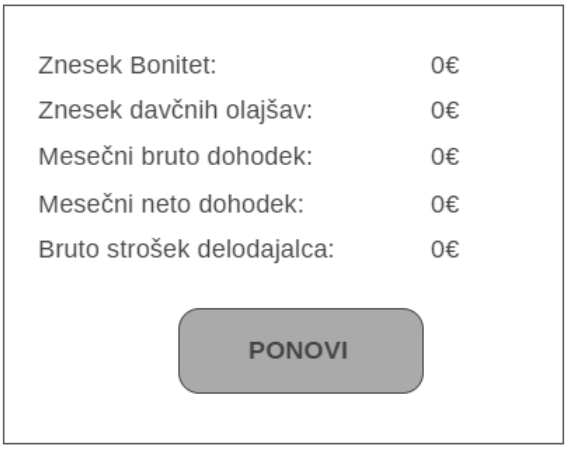
3.3.1 Spletni vmesnik

Spletni vmesnik smo napisali s pomočjo programskega jezika PHP in jezika HTML [23, 16]. Za oblikovanje smo uporabili ogrodje Bootstrap [2]. Spletni vmesnik je sestavljen iz več vnosnih polj, in sicer iz polja za vnos plače, spustnega seznama za izbor vrste plače (bruto, neto, strošek delodajalca), polja za vnos števila vzdrževanih otrok, polja za vnos števila vzdrževanih otrok, ki potrebujejo posebno nego, polja za vnos bonitet, potrditvenega okvirčka za izbiro 100% invalidnost in potrditvenega gumba `Izračunaj`. Načrt prvega dela lahko vidimo na skici 3.3.



Slika 3.3: Skica prvega dela spletnega vmesnika

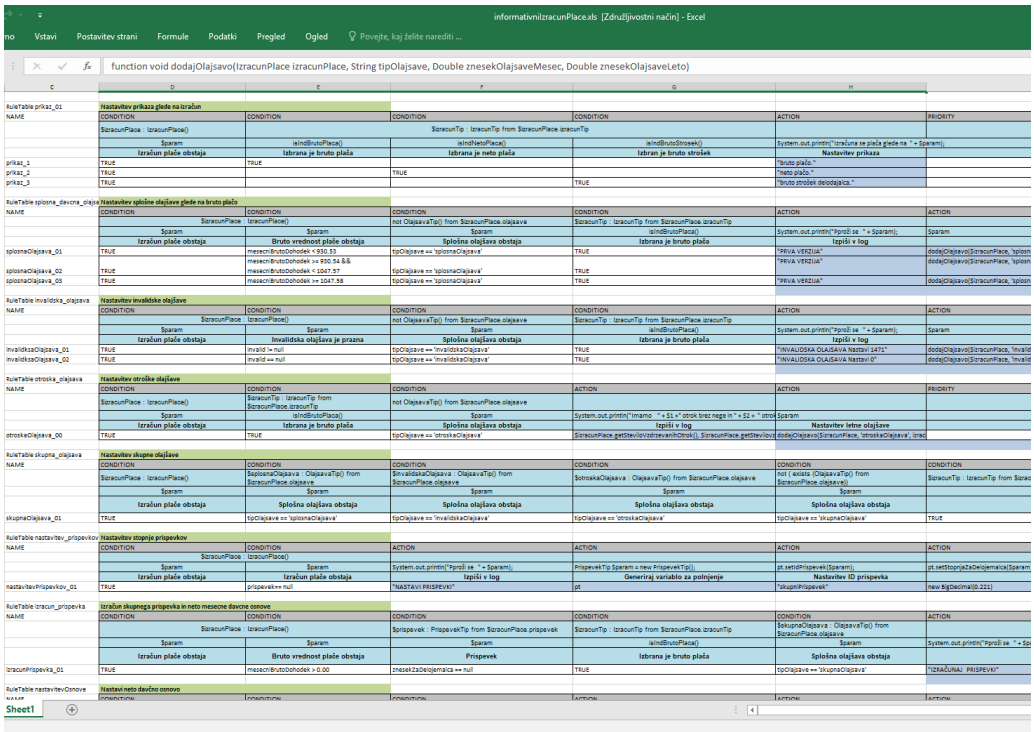
Vsi podatki iz vnosnih polj se potem sestavijo v JSON zahtevek in pošljejo na KIE izvedbeni strežnik. Ko KIE izvedbeni strežnik vrne odgovor, se odpre drugi del spletnega vmesnika s rezultati in sicer znesek bonitet, znesek davčnih olajšav, mesečni bruto dohodek, mesečni neto dohodek, bruto strošek delodajalca in na koncu še gumb za vrnitev na začetek izračuna. Načrt drugega dela lahko vidimo na skici 3.4.



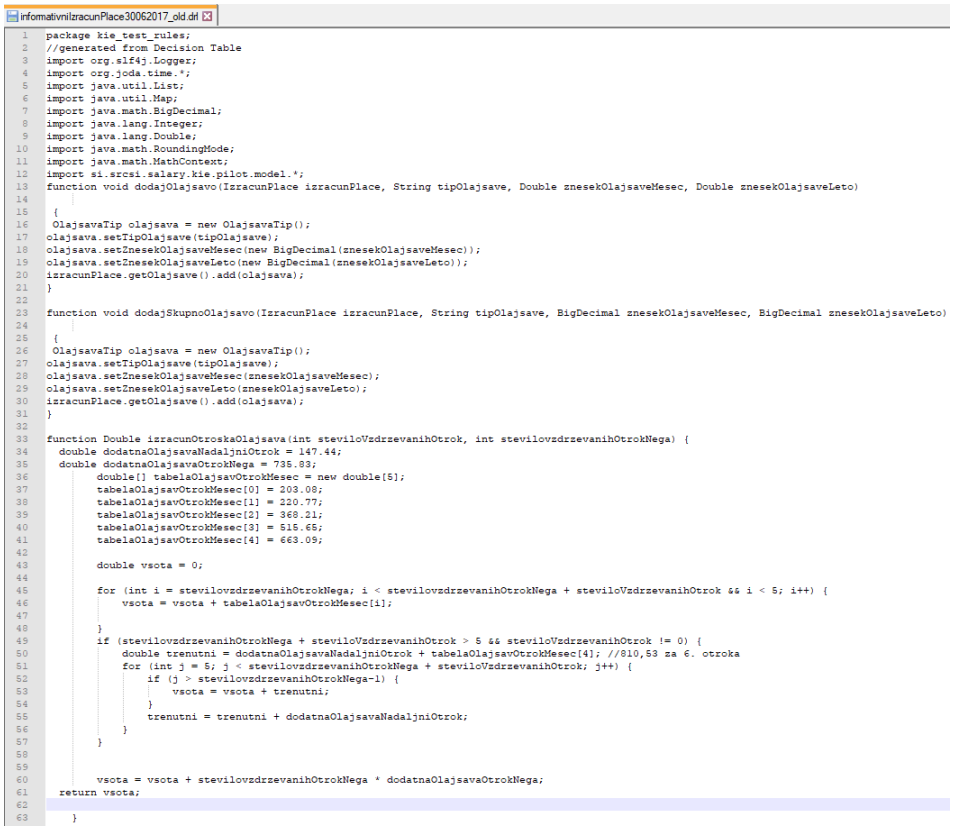
Slika 3.4: Skica drugega dela spletnega vmesnika

3.4 Microsoft Excel

Microsoft Excel, je program iz nabora programov Microsoft Office, ki je program za obdelavo podatkov v razpredelnicah [21]. Zaradi preprostosti uporabe in preglednosti smo ga uporabili pri projektu za pisanje pravil. Za delovanje platforme Drools sicer ni nujno potreben, izkušnje kažejo, da ga predvsem analitiki uporabljajo za pisanje pravil. Zato smo se odločili, da prikažemo primer v programu Microsoft Excel. V sistemu Drools se pišejo pravila v njihovem programskem jeziku DRL, ki je zelo podoben jeziku Java. Ta jezik lahko uporabimo v programu Excel, ki se potem prevede v `.drl` datoteko. Primer pravil zapisanih v programu Microsoft Excel lahko vidimo na sliki 3.5, primer pravil v DRL jeziku pa na sliki 3.6



Slika 3.5: Pravila za informativni izračun plače



infomativnIzracunPlace30062017_old.drl

```
package kie_test_rules;
//generated from Decision Table
import org.slf4j.Logger;
import org.joda.time.*;
import java.util.List;
import java.util.Map;
import java.math.BigDecimal;
import java.lang.Integer;
import java.lang.Double;
import java.math.RoundingMode;
import java.math.MathContext;
import si.srcsi.salary.kie.pilot.model.*;
function void dodajOlajsavo(IzracunPlace izracunPlace, String tipOlajsave, Double znesekOlajsaveMesec, Double znesekOlajsaveLeto)

 {
 OlajsavaTip olajsava = new OlajsavaTip();
olajsava.setTipOlajsave(tipOlajsave);
olajsava.setZnesekOlajsaveMesec(new BigDecimal(znesekOlajsaveMesec));
olajsava.setZnesekOlajsaveLeto(new BigDecimal(znesekOlajsaveLeto));
izracunPlace.getOlajsave().add(olajsava);
}

function void dodajSkupnoOlajsavo(IzracunPlace izracunPlace, String tipOlajsave, BigDecimal znesekOlajsaveMesec, BigDecimal znesekOlajsaveLeto)

 {
 OlajsavaTip olajsava = new OlajsavaTip();
olajsava.setTipOlajsave(tipOlajsave);
olajsava.setZnesekOlajsaveMesec(znesekOlajsaveMesec);
olajsava.setZnesekOlajsaveLeto(znesekOlajsaveLeto);
izracunPlace.getOlajsave().add(olajsava);
}

function Double izracunOtroskaOlajsava(int steviloVzdrzevanihOtrok, int stevilovzdrzevanihOtrokNega) {
  double dodatnaOlajsavaNadaljniOtrok = 147.44;
  double dodatnaOlajsavaOtrokNega = 735.93;
        double[] tabelaOlajsavOtrokMesec = new double[5];
        tabelaOlajsavOtrokMesec[0] = 203.08;
        tabelaOlajsavOtrokMesec[1] = 220.77;
        tabelaOlajsavOtrokMesec[2] = 368.21;
        tabelaOlajsavOtrokMesec[3] = 515.65;
        tabelaOlajsavOtrokMesec[4] = 663.09;

        double vsota = 0;

        for (int i = stevilovzdrzevanihOtrokNega; i < stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok && i < 5; i++) {
            vsota = vsota + tabelaOlajsavOtrokMesec[i];

        }
        if (stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok > 5 && steviloVzdrzevanihOtrok != 0) {
            double trenutni = dodatnaOlajsavaNadaljniOtrok + tabelaOlajsavOtrokMesec[4]; //810,53 za 6. otroka
            for (int j = 5; j < stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok; j++) {
                if (j > stevilovzdrzevanihOtrokNega-1) {
                    vsota = vsota + trenutni;
                }
                trenutni = trenutni + dodatnaOlajsavaNadaljniOtrok;
            }
        }


        vsota = vsota + stevilovzdrzevanihOtrokNega * dodatnaOlajsavaOtrokNega;
 return vsota;

   }
```

Slika 3.6: Prvi del prevedenih pravil v DRL

```
// rule values at C19, header at C14
rule "splosnaOlajsava_01"
    salience 80
    no-loop true
    when
        $izracunPlace : IzracunPlace(true, mesecniBrutoDohodek < 930.53)
        not OlajsavaTip(tipOlajsave == 'splosnaOlajsava') from $izracunPlace.olajsave
    //  $izracunTip : IzracunTip(isIndBrutoPlaca() == "true") from $izracunPlace.izracunTip
    then
        System.out.println("////// 5 //////");
        dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 543.32, 6519.82);
        update($izracunPlace);

end

// rule values at C20, header at C14
rule "splosnaOlajsava_02"
    salience 80
    no-loop true
    when
        $izracunPlace : IzracunPlace(true, mesecniBrutoDohodek >= 930.54 && mesecniBrutoDohodek < 1047.57)
        not OlajsavaTip(tipOlajsave == 'splosnaOlajsava') from $izracunPlace.olajsave
    //  $izracunTip : IzracunTip(isIndBrutoPlaca() == "true") from $izracunPlace.izracunTip
    then
        System.out.println("////// 6 //////");
        dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 368.22, 4418.64);
        update($izracunPlace);

end

// rule values at C21, header at C14
rule "splosnaOlajsava_03"
    salience 80
    no-loop true
    when
        $izracunPlace : IzracunPlace(true, mesecniBrutoDohodek >= 1047.58 || mesecniBrutoDohodek == null)
        not OlajsavaTip(tipOlajsave == 'splosnaOlajsava') from $izracunPlace.olajsave
        //$izracunTip : IzracunTip(isIndBrutoPlaca() == "true") from $izracunPlace.izracunTip
    then
        System.out.println("////// 7 //////");
        dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 275.22, 3302.70);
        update($izracunPlace);
end

// rule values at C28, header at C23
rule "invalidksaOlajsava_01"
    salience 80
    no-loop true
    when
        $izracunPlace : IzracunPlace(true, invalid != null)
        not OlajsavaTip(tipOlajsave == 'invalidskaOlajsava') from $izracunPlace.olajsave
    //  $izracunTip : IzracunTip(isIndBrutoPlaca() == "true") from $izracunPlace.izracunTip
    then
        System.out.println("////// 8 //////");
        //System.out.println("Pproži se    " + "INVALIDSKA OLAJSAVA Nastavi 1471");
        dodajOlajsavo($izracunPlace, 'invalidskaOlajsava', 1471.57, 17658.84);
        update($izracunPlace);
end
```

Slika 3.7: Drugi del prevedenih pravil v DRL

Poglavje 4

Prikaz uporabe na primeru izračuna plač

Ker bomo našo spletno aplikacijo primerjali s že obstoječo aplikacijo, mora le-ta pokrivati iste funkcionalne in nefunkcionalne zahteve, ki jih bomo v nadaljevanju predstavili.

V drugem delu bo predstavljeno delovanje naše spletne aplikacije, katera upodablja platformo Drools. Prikazan bo tudi primer, ki izpostavi vsa pravila, ki se izvršijo ob določenem primeru, kaj je njihov pomen ter kako poteka postopek spremembe poslovnih pravil.

4.1 Zahteve aplikacije

4.1.1 Funkcionalne zahteve

- Izračun bruto plače iz neto plače ali iz stroška delodajalca
  - Spletna aplikacija mora v primeru vnosa enega izmed podatkov (neto plače ali pa stroška delodajalca) izračunati bruto plačo.
- Izračun neto plače iz bruto plače ali iz stroška delodajalca

  - Spletna aplikacija mora v primeru vnosa enega izmed podatkov (bruto plače ali pa stroška delodajalca) izračunati neto plačo.

- Izračun stroška delodajalca iz neto ali bruto plače

  - Spletna aplikacija mora v primeru vnosa enega izmed podatkov (bruto plače ali pa neto plače) izračunati bruto strošek delodajalca.

- Upoštevanje olajšav za otroke in otroke z posebno nego

  - Spletna aplikacija mora v primeru vnosa olajšav za otroke ali otroke, ki potrebujejo posebno nego, upoštevati to pri izračunu rezultata.

- Upoštevanje osebne olajšave 100% invalidnosti

  - Spletna aplikacija mora v primeru izbire osebne olajšave 100% invalidnost to upoštevati pri izračunu rezultata.

- Validacija vnosnih polj

  - Spletna aplikacija ne sme uporabniku dovoliti vnos črk, ampak le številke. Aplikacija mora tudi preverjati, da je izpolnjeno vnosno polje za znesek plače, ker je obvezen podatek.

- Omogočeni preprosti popravki ob spremembi zakonov

  - Spletna aplikacija mora biti napisana tako, da omogoča čim lažje spreminjanje pogojev.

4.1.2 Nefunkcionalne zahteve

- Preprost in razumljiv uporabniški vmesnik

  - Uporabniški vmesnik mora imeti logično postavljena vnosna polja in biti dovolj preprost za uporabo, da ne zmede uporabnika.

- Možnost izbire vrste plače (neto, bruto in strošek delodajalca)
  - Uporabnik mora imeti možnost izbire za kateri znesek se gre (neto, bruto ali strošek delodajalca).
- Vnos raznih olajšav (invalidnost, otroci, bonitete)
  - Uporabniški vmesnik mora ponujati uporabniku vnos števila otrok in otrok, ki potrebujejo posebno nego ter višino bonitet. Poleg tega mora še ponujati, da lahko uporabnik označi v primeru, da ima še posebno olajšavo 100% invalidnost.

4.2 Delovanje aplikacije

Da omogočimo delovanje naše aplikacije, moramo prvo zagnati program Docker in postaviti prej opisane Docker vsebnike. Naša poslovna pravila morajo biti poleg docker vsebnika v mapi s konfiguracijo. To storimo tako da se preko ukaznega poziva premaknemo v direktorij s konfiguracijo za izbran Docker vsebnik in s ukazom
‘‘`docker build -t src-rules-execution-infoizracun-container .`’’
zgradimo docker vsebnik in glede na konfiguracijo se pripravijo vse potrebne datoteke. Isto storimo za vsebnik s čelno aplikacijo s podobnim ukazom. Nato s ukazom ‘‘`docker run --privileged=true --net javaapp -p 4000:8080 --name src-rules-execution-infoizracun -i -t src-rules-execution-infoizracun-container`’’ postavimo vsebnik.

Ko imamo oba vsebnika postavljena, lahko preko brskalnika dostopamo do naše spletne aplikacije na vratih 8080. Primer kalkulatorja plač se na prvi pogled zdi kot preprosta aplikacija. Kot vidimo spodaj na slikah 4.1 in 4.2 je čelni del preprosto sestavljen iz treh vnosnih polj, enega spustnega seznama in enega potrditvenega polja. Obvezen podatek je znesek plače. Ko vnesemo željene podatke, s pritiskom na gumb `Izračunaj` sprožimo izračun plače in olajšav glede na vnesene podatke.

Slika 4.1: Čelni del spletne aplikacije

Aplikacija sestavi JSON zahtevek iz naših vnesenih podatkov in ga pošlje na KIE izvedbeni strežnik. Ta obdela naše podatke in sprožijo se pravila glede na naše vnešene podatke. Rezultate sproženih pravil sestavi v JSON odgovor in ga vrne naši spletni aplikaciji, ki nato na zaslon izpiše rezultat - to je neto plača, bruto plača, bruto strošek delodajalca, znesek bonitet in znesek davčnih olajšav. V naslednjem podpoglavju bomo predstavili katera poslovna pravila se sprožijo za primer izračuna s podatkom 1500 EUR bruto plače.

V primeru, da želimo ponoviti oziroma znova začeti izračun, pritisnemo na gumb `Ponovi` in zopet lahko vnesemo poljubne podatke. Pri spletni aplikaciji je tudi urejeno preverjanje, da se lahko vnaša samo numerične znake in da polja ne smejo ostati prazna. Aplikacija nam omogoča različne izračune: iz neto plače v bruto plačo in obratno ter izračun bruto stroška delodajalca, kjer seveda upošteva tudi invalidnost, število otrok in znesek bonitet.

Slika 4.2: Čelni del spletne aplikacije - izračun

4.3 Delovanje poslovnih pravil

Del uporabljene Excel datoteke s pravili vidimo na sliki 4.3. V prvi vrstici je naslov množice pravil (angl. RuleSet). Sledeča vrstica je rezervirana za uvoze (angl. imports), kjer lahko uporabljamo uvoze iz programskega jezika Java. Sledi vrstica rezervirana za morebitne posebne funkcije, ki jih lahko kličemo v izvedbenem delu pravila.

RuleSet	kie test rules			
import	org.slf4j.Logger, org.joda.time.*, java.util.List, java.util.Map, java.math.BigDecimal, java.lang.Integer, java.lang.Double, java.math.RoundingMode, java.math.MathContext, si.srcsi.salary.kie.pilot.model.*			

Functions	Double znesekOlajsaveMesec, Double znesekOlajsaveLeto)
	{
	OlajsavaTip olajsava = new OlajsavaTip();
	olajsava.setTipOlajsave(tipOlajsave);
	olajsava.setZnesekOlajsaveMesec(new BigDecimal(znesekOlajsaveMesec));
	olajsava.setZnesekOlajsaveLeto(new BigDecimal(znesekOlajsaveLeto));
	izracunPlace.getOlajsave().add(olajsava);
	}
	function void dodajSkupnoOlajsavo(IzracunPlace izracunPlace,
	String tipOlajsave, BigDecimal znesekOlajsaveMesec, BigDecimal znesekOlajsaveLeto)
	{
	OlajsavaTip olajsava = new OlajsavaTip();
	olajsava.setTipOlajsave(tipOlajsave);
	olajsava.setZnesekOlajsaveMesec(znesekOlajsaveMesec);
	olajsava.setZnesekOlajsaveLeto(znesekOlajsaveLeto);
	izracunPlace.getOlajsave().add(olajsava);
	}
	function Double izracunOtroskaOlajsava(int steviloVzdrzevanihOtrok, int steviloVzdrzevanihOtrokNega) {
	double dodatnaOlajsavaNadaljniOtrok = 147.44;
	double dodatnaOlajsavaOtrokNega = 735.83;
	double[] tabelaOlajsavOtrokMesec = new double[5];
	tabelaOlajsavOtrokMesec[0] = 203.08;
	tabelaOlajsavOtrokMesec[1] = 220.77;
	tabelaOlajsavOtrokMesec[2] = 368.21;
	tabelaOlajsavOtrokMesec[3] = 515.65;
	tabelaOlajsavOtrokMesec[4] = 663.09;
	double vsota = 0;			

RuleTable prikaz_01	Nastavitev prikaza glede na izračun			
NAME	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		$izracunTip : IzracunTip from $izracunPlace.izracunTip	
	$param	isIndBrutoPlaca()	isIndNetoPlaca()	isIn
	Izračun plače obstaja	Izbrana je bruto plača	Izbrana je neto plača	Izbran
prikaz_1	TRUE	TRUE		
prikaz_2	TRUE		TRUE	
prikaz_3	TRUE			TRUE
RuleTable splosna_davcna_olajsa	Nastavitev splošne olajšave glede na bruto plačo			
NAME	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		not OlajsavaTip() from $izracunPlace.olajsave	$izracunTip : IzracunTip from $izracunPlace.izracu
	$param	$param	$param	isI
	Izračun plače obstaja	Bruto vrednost plače obstaja	Splošna olajšava obstaja	Izbra
splosnaOlajsava_01	TRUE	mesecniBrutoDohodek < 930.53	tipOlajsave == 'splosnaOlajsava'	TRUE
splosnaOlajsava_02	TRUE	mesecniBrutoDohodek >= 930.54 && mesecniBrutoDohodek < 1047.57	tipOlajsave == 'splosnaOlajsava'	TRUE
splosnaOlajsava_03	TRUE	mesecniBrutoDohodek >= 1047.58	tipOlajsave == 'splosnaOlajsava'	TRUE

Slika 4.3: Del poslovnih pravil

V našem primeru imamo `dodajOlajsavo`, `dodajSkupnoOlajsavo`, `izracunOtroskaOlajsava`, ki jih vidimo na sliki 4.4. Nato sledijo naša poslovna pravila. Vseh pravil v Excel datoteki je 21. V našem izbranem primeru se jih sproži 10 in v nadaljevanju se bomo osredotočili le na ta. Ko KIE izvedbeni strežnik prejme JSON zahtevek, se začne preverjanje pogojev za vsako pravilo posebej in sicer v enakem vrstnem redu, kot jih vidimo v Excel datoteki. Za prikaz delovanja, bomo vnesli primer s bruto plačo 1500 EUR.

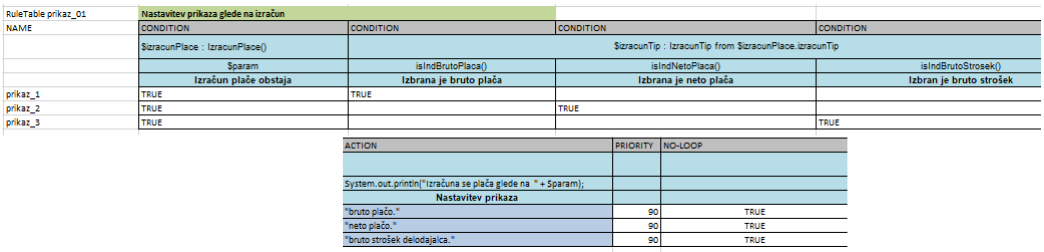
V nadaljevanju bomo vsako od sproženih pravil v izbranem primeru pokazali in opisali njihovo delovanje.

```
function void dodajOlajsavo(IzracunPlace izracunPlace, String tipOlajsave, Double znesekOlajsaveMesec, Double znesekOlajsaveLeto)

{
OlajsavaTip olajsava = new OlajsavaTip();
olajsava.setTipOlajsave(tipOlajsave);
olajsava.setZnesekOlajsaveMesec(new BigDecimal(znesekOlajsaveMesec));
olajsava.setZnesekOlajsaveLeto(new BigDecimal(znesekOlajsaveLeto));
izracunPlace.getOlajsave().add(olajsava);
}

function void dodajSkupnoOlajsavo(IzracunPlace izracunPlace, String tipOlajsave, BigDecimal znesekOlajsaveMesec, BigDecimal znesekOlajsaveLeto)

{
OlajsavaTip olajsava = new OlajsavaTip();
olajsava.setTipOlajsave(tipOlajsave);
olajsava.setZnesekOlajsaveMesec(znesekOlajsaveMesec);
olajsava.setZnesekOlajsaveLeto(znesekOlajsaveLeto);
izracunPlace.getOlajsave().add(olajsava);
}



function Double izracunOtroskaOlajsava(int steviloVzdrzevanihOtrok, int stevilovzdrzevanihOtrokNega) {
 double dodatnaOlajsavaNadaljniOtrok = 147.44;
 double dodatnaOlajsavaOtrokNega = 735.83;
    double[] tabelaOlajsavOtrokMesec = new double[5];
    tabelaOlajsavOtrokMesec[0] = 203.08;
    tabelaOlajsavOtrokMesec[1] = 220.77;
    tabelaOlajsavOtrokMesec[2] = 368.21;
    tabelaOlajsavOtrokMesec[3] = 515.65;
    tabelaOlajsavOtrokMesec[4] = 663.09;

    double vsota = 0;

    for (int i = stevilovzdrzevanihOtrokNega; i < stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok && i < 5; i++) {
      vsota = vsota + tabelaOlajsavOtrokMesec[i];

    }
    if (stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok > 5 && steviloVzdrzevanihOtrok != 0) {
      double trenutni = dodatnaOlajsavaNadaljniOtrok + tabelaOlajsavOtrokMesec[4]; //810,53 za 6. otroka
      for (int j = 5; j < stevilovzdrzevanihOtrokNega + steviloVzdrzevanihOtrok; j++) {
        if (j > stevilovzdrzevanihOtrokNega-1) {
          vsota = vsota + trenutni;
        }
        trenutni = trenutni + dodatnaOlajsavaNadaljniOtrok;
      }
    }


    vsota = vsota + stevilovzdrzevanihOtrokNega * dodatnaOlajsavaOtrokNega;
return vsota;

 }
```

Slika 4.4: Funkcije

4.3.1 Prikaz delovanja pravil na primeru

Za prikaz delovanja bomo vnesli primer s bruto plačo 1500 EUR. Vsako od sproženih pravil v našem izbranem primeru bomo pokazali in opisali njihovo delovanje. Pravilo `prikaz_1` na sliki 4.5 nastavi, katero vrsto plače smo izbrali. V našem primeru je to bruto plača.

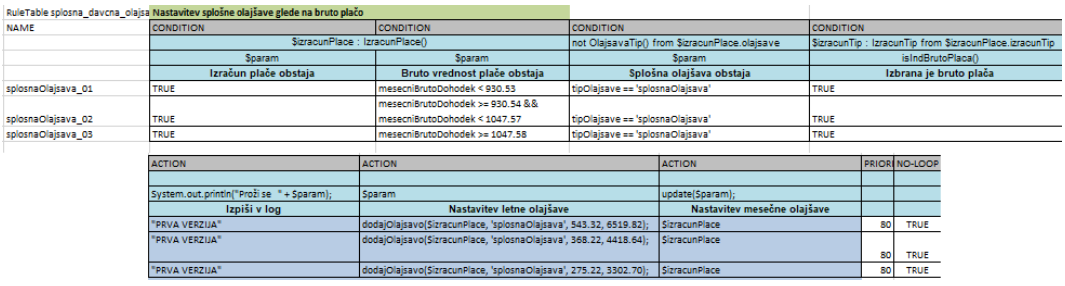


RuleTable prikaz_01	Nastavitev prikaza glede na izračun			
NAME	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()	$izracunTip : IzracunTip from $izracunPlace.izracunTip		
	$param	isIndBrutoPlaca()	isIndNetoPlaca()	isIndBrutoStrosek()
	Izračun plače obstaja	Izbrana je bruto plača	Izbrana je neto plača	Izbran je bruto strošek
prikaz_1	TRUE	TRUE		
prikaz_2	TRUE		TRUE	
prikaz_3	TRUE			TRUE

ACTION	PRIORITY	NO-LOOP
System.out.println("Izračuna se plača glede na " + $param);		
Nastavitev prikaza		
"bruto plačo,"	90	TRUE
"neto plačo,"	90	TRUE
"bruto strošek delodajalca,"	90	TRUE

Slika 4.5: Pravilo za nastavitev vrste izračuna

Nato se sproži pravilo `splosnaOlajsava_03`, ker smo vnesli mesečno bruto plačo višjo od 1047,58 EUR in nam nastavi splošno olajšavo na 275,22 EUR mesečno oziroma 3302,70 EUR letno. Pravilo vidimo na sliki 4.6.



RuleTable splosna_davcna_olajsa	Nastavitev splošne olajšave glede na bruto plačo			
NAME	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		not OlajsavaTip() from $izracunPlace.olajsave	$izracunTip : IzracunTip from $izracunPlace.izracunTip
	$param	$param	$param	isIndBrutoPlaca()
	Izračun plače obstaja	Bruto vrednost plače obstaja	Splošna olajšava obstaja	Izbrana je bruto plača
splosnaOlajsava_01	TRUE	mesecniBrutoDohodek < 930.53	tipOlajsave == 'splosnaOlajsava'	TRUE
splosnaOlajsava_02	TRUE	mesecniBrutoDohodek >= 930.54 && mesecniBrutoDohodek < 1047.57	tipOlajsave == 'splosnaOlajsava'	TRUE
splosnaOlajsava_03	TRUE	mesecniBrutoDohodek >= 1047.58	tipOlajsave == 'splosnaOlajsava'	TRUE

ACTION	ACTION	ACTION	PRIOR	NO-LOOP
System.out.println("Proži se " + $param);	$param	update($param);		
Izpiši v log	Nastavitev letne olajšave	Nastavitev mesečne olajšave		
"PRVA VERZIJA"	dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 543.32, 6519.82);	$izracunPlace	80	TRUE
"PRVA VERZIJA"	dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 368.22, 4418.64);	$izracunPlace	80	TRUE
"PRVA VERZIJA"	dodajOlajsavo($izracunPlace, 'splosnaOlajsava', 275.22, 3302.70);	$izracunPlace	80	TRUE

Slika 4.6: Pravilo za nastavitev splošne olajšave

Ker nismo izbrali osebne davčne olajšave za 100% invalidnost, je naslednje sproženo pravilo `invalidksaOlajsava_02` vidno na sliki 4.7. To pravilo nastavi, da nimamo invalidske olajšave, kar pomeni, da jo nastavi na 0.00 EUR. V primeru, če bi imeli olajšavo za 100% invalidnost bi pa to pravilo nastavilo olajšavo na 1471.71 EUR mesečno oziroma 17658.84 EUR letno.

RuleTable invalidska_olajsava	Nastavitev invalidske olajšave			
NAME	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		not OlajsavaTip() from $izracunPlace.olajsave	$izracunTip : IzracunTip from $izracunPlace.izracunTip
	$param	$param	$param	isIndBrutoPlaca()
	Izračun plače obstaja	Invalidska olajšava je prazna	Splošna olajšava obstaja	Izbrana je bruto plača
invalidksaOlajsava_01	TRUE	invalid != null	tipOlajsave == 'invalidskaOlajsava'	TRUE
invalidksaOlajsava_02	TRUE	invalid == null	tipOlajsave == 'invalidskaOlajsava'	TRUE

ACTION	ACTION	ACTION	PRIOR	NO-LOOP
System.out.println("Proši se " + $param);	$param	update($param);		
Izpiši v log	Nastavitev letne olajšave	Nastavitev mesečne olajšave		
"INVALIDSKA OLAJSAVA Nastavi 1471"	dodajOlajsavo($izracunPlace, 'invalidskaOlajsava', 1471.57, 17658.8	$izracunPlace	80	TRUE
"INVALIDSKA OLAJSAVA Nastavi 0"	dodajOlajsavo($izracunPlace, 'invalidskaOlajsava', 0.00, 0.00);	$izracunPlace	80	TRUE

Slika 4.7: Pravilo za nastavitev invalidske olajšave

Sledi pravilo `otroskaOlajsava_00`, ki nam izračuna višino otroške olajšave in je vidno na sliki 4.8. V našem primeru nismo vnesli otrok, zato je 0.00 EUR. Sicer se izračuna v funkciji `izracunOtroskaOlajsava` v vrstici `functions`, ki je vidna na sliki 4.9.

RuleTable otroska_olajsava	Nastavitev otroške olajšave			
NAME	CONDITION	CONDITION	CONDITION	ACTION
	$izracunPlace : IzracunPlace()	$izracunTip : IzracunTip from $izracunPlace.izracunTip	not OlajsavaTip() from $izracunPlace.olajsave	
	$param	isIndBrutoPlaca()	$param	System.out.println("imamo " + $1 +" otrok brez nege in " + $2 + " otrok ki potrebujejo nego. Proši se pravilo:" + $3);
	Izračun plače obstaja	Izbrana je bruto plača	Splošna olajšava obstaja	Izpiši v log
otroskaOlajsava_00	TRUE	TRUE	tipOlajsave == 'otroskaOlajsava'	$izracunPlace.getSteviloVzdrzevanihOtrok(), $izracunPlace.getSteviloVzdrzevanihOtrokNega(),0

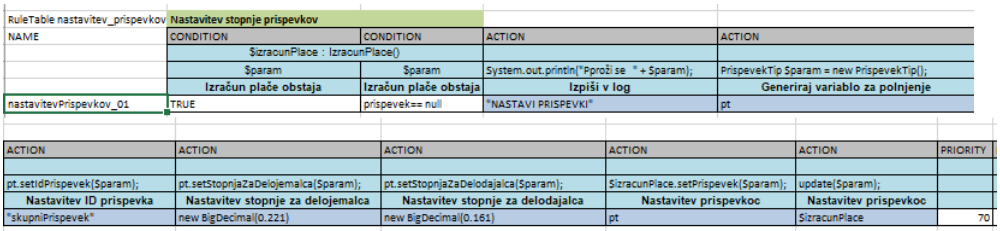
ACTION	PRIOR
$param	
Nastavitev letne olajšave	
dodajOlajsavo($izracunPlace, 'otroskaOlajsava', izracunOtroskaOlajsava($izracunPlace.getSteviloVzdrzevanihOtrok(), $izracunPlace.getSteviloVzdrzevanihOtrokNega()), izracunOtroskaOlajsava($izracunPlace.getSteviloVzdrzevanihOtrok(), $izracunPlace.getSteviloVzdrzevanihOtrokNega()) *12);	80

Slika 4.8: Pravilo za nastavitev otroške olajšave

```
function Double izracunOtroskaOlajsava(int steviloVzdrzevanihOtrok, int steviloVzdrzevanihOtrokNega) {
 double dodatnaOlajsavaNadaljniOtrok = 147.44;
 double dodatnaOlajsavaOtrokNega = 735.83;
    double[] tabelaOlajsavOtrokMesec = new double[5];
    tabelaOlajsavOtrokMesec[0] = 203.08;
    tabelaOlajsavOtrokMesec[1] = 220.77;
    tabelaOlajsavOtrokMesec[2] = 368.21;
    tabelaOlajsavOtrokMesec[3] = 515.65;
    tabelaOlajsavOtrokMesec[4] = 663.09;

    double vsota = 0;

    for (int i = steviloVzdrzevanihOtrokNega; i < steviloVzdrzevanihOtrokNega + steviloVzdrzevanihOtrok && i < 5; i++) {
      vsota = vsota + tabelaOlajsavOtrokMesec[i];

    }
    if (steviloVzdrzevanihOtrokNega + steviloVzdrzevanihOtrok > 5 && steviloVzdrzevanihOtrok != 0) {
      double trenutni = dodatnaOlajsavaNadaljniOtrok + tabelaOlajsavOtrokMesec[4]; //810,53 za 6. otroka
      for (int j = 5; j < steviloVzdrzevanihOtrokNega + steviloVzdrzevanihOtrok; j++) {
        if (j > steviloVzdrzevanihOtrokNega-1) {
          vsota = vsota + trenutni;
        }
        trenutni = trenutni + dodatnaOlajsavaNadaljniOtrok;
      }
    }
    vsota = vsota + steviloVzdrzevanihOtrokNega * dodatnaOlajsavaOtrokNega;
return vsota;

 }
```

Slika 4.9: Funkcija za izračun otroške olajšave

Nato se sproži pravilo `nastavitevPrispevkov_01`, ki je vidno na sliki 4.10. To nastavi stopnje prispevkov in sicer 0.221 za delojemalca in 0.161 za delodajalca.

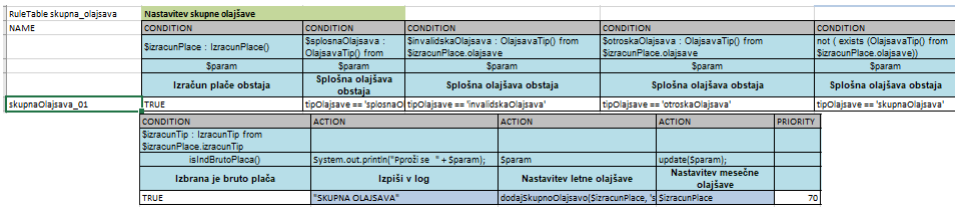


RuleTable nastavitev_prispevkov	Nastavitev stopnje prispevkov			
NAME	CONDITION	CONDITION	ACTION	ACTION
	$izracunPlace : IzracunPlace()			
	$param	$param	System.out.println("Proži se " + $param);	PrispevekTip $param = new PrispevekTip();
	Izračun plače obstaja	**Izračun plače obstaja**	**Izpiši v log**	**Generiraj variablo za polnjenje**
nastavitevPrispevkov_01	TRUE	prispevek== null	"NASTAVI PRISPEVKI"	pt

ACTION	ACTION	ACTION	ACTION	ACTION	PRIORITY
pt.setIdPrispevek($param);	pt.setStopnjaZaDelojemalca($param);	pt.setStopnjaZaDelodajalca($param);	$izracunPlace.setPrispevek($param);	update($param);	
Nastavitev ID prispevka	**Nastavitev stopnje za delojemalca**	**Nastavitev stopnje za delodajalca**	**Nastavitev prispevkoc**	**Nastavitev prispevkoc**	
"skupniPrispevek"	new BigDecimal(0.221)	new BigDecimal(0.161)	pt	$izracunPlace	70

Slika 4.10: Pravilo za nastavitev stopnje prispevkov

Na sliki 4.11 vidimo pravilo `skupnaOlajsava_01`, ki doda vrednosti vseh olajšav (invalidska, otroška, splošna) v skupno olajšavo.



RuleTable skupna_olajsava	Nastavitev skupne olajšave				
NAME	CONDITION	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()	$splosnaOlajsava : OlajsavaTip() from	$invalidskaOlajsava : OlajsavaTip() from $izracunPlace.olajsave	$otroskaOlajsava : OlajsavaTip() from $izracunPlace.olajsave	not (exists (OlajsavaTip() from $izracunPlace.olajsave))
	$param	$param	$param	$param	$param
	Izračun plače obstaja	**Splošna olajšava obstaja**	**Splošna olajšava obstaja**	**Splošna olajšava obstaja**	**Splošna olajšava obstaja**
skupnaOlajsava_01	TRUE	tipOlajsave == 'splosnaO	tipOlajsave == 'invalidskaOlajsava'	tipOlajsave == 'otroskaOlajsava'	tipOlajsave == 'skupnaOlajsava'

CONDITION	ACTION	ACTION	ACTION	PRIORITY
$izracunTip : IzracunTip from $izracunPlace.izracunTip				
isIndBrutoPlaca()	System.out.println("Proži se " + $param);	$param	update($param);	
Izbrana je bruto plača	**Izpiši v log**	**Nastavitev letne olajšave**	**Nastavitev mesečne olajšave**	
TRUE	"SKUPNA OLAJSAVA"	dodajSkupnoOlajsavo($izracunPlace, '$	$izracunPlace	70

Slika 4.11: Pravilo za nastavitev skupne olajšave

Sledeče pravilo, ki se sproži je `izracunPrispevka_01` viden na sliki 4.12 in 4.13. To pravilo opravi izračun skupnega prispevka in neto mesečne davčne osnove in sicer znesek za delojemalca izračuna po formuli :

`(stopnjaZaDelojemalca*mesečniBrutoDohodek)+znesekBonitet`

za delodajalca pa :

`(stopnjaZaDelodajalca*mesečniBrutoDohodek)+znesekBonitet`

Izračuna tudi neto mesečno davčno osnovo po formuli :

`(mesečniBrutoDohodek-(znesekZaDelojemalca+ZnesekOlajšaveMesec))+znesekBonitet`

RuleTable izracun_prispevka	Izračun skupnega prispevka in neto mesecne davcne osnove				
NAME	CONDITION	CONDITION	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		$prispevek : PrispevekTip from $izracunPlace.prispevek	$izracunTip : IzracunTip from $izracunPlace.izracunTip	$skupnaOlajsava : OlajsavaTip() from
	$param	$param	$param	isIndBrutoPlaca()	$param
	Izračun plače obstaja	Bruto vrednost plače obstaja	Prispevek	Izbrana je bruto plača	Splošna olajšava obstaja
izracunPrispevka_01	TRUE	mesecniBrutoDohodek > 0.00	znesekZaDelojemalca == null	TRUE	tipOlajsave == 'skupnaOlajsava'

ACTION	ACTION
System.out.println("Sprozi se " + $param);	$prispevek.setZnesekZaDelojemalca($param);
Izpiši v log	Nastavitev stopnje za delojemalca
"IZRAČUNAJ PRISPEVKI"	$prispevek.getStopnjaZaDelojemalca().multiply($izracunPlace.getMesecniBrutoDohodek().add($izracunPlace.getZnesekBonitet()))

Slika 4.12: Pravilo za izračun skupnega prispevka in mesečne davčne osnove - 1. del

ACTION
$prispevek.setZnesekZaDelodajalca($param);
Nastavitev stopnje za delodajalca
$prispevek.getStopnjaZaDelodajalca().multiply($izracunPlace.getMesecniBrutoDohodek().add($izracunPlace.getZnesekBonitet()))

ACTION	ACTION	PRIORITY
$izracunPlace.setNetoMesecnaDavcnaOsnova($param);	update($param);	
Izračun davčne osnove	Nastavitev mesečne	
($izracunPlace.getMesecniBrutoDohodek().subtract($prispevek.getZnesekZaDelojemalca().add($skupnaOlajsava.getZnesekOlajsaveMesec()))).add($izracunPlace.getZnesekBonitet())	$izracunPlace	60

Slika 4.13: Pravilo za izračun skupnega prispevka in mesečne davčne osnove - 2. del

Po tem se iz nabora pravil za izračun dohodnine sproži pravilo `izracunDohodnine_03`, ki ga vidimo na sliki 4.14. Sporži se, ker je `netoMesecnaDavcnaOsnova` večja od 668.44 EUR in manjša od 1700 EUR. Nato nastavi `znesekDohodnine` in sicer po sledeči formuli :

```
(106.95+NetoMesecnaDavcnaOsnova-668.44)*0.27
```

RuleTable izracun_dohodnine	Izračun dohodnine		
NAME	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		
	$param	$param	$param
	Izračun plače obstaja	Bruto vrednost plače obstaja	Neto davčna osnova
izracunDohodnine_01	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova < 0
izracunDohodnine_02	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova > 0 && netoMesecnaDavcnaOsnova <= 668.44
izracunDohodnine_03	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova > 668.44 && netoMesecnaDavcnaOsnova <= 1700.00
izracunDohodnine_04	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova > 1700.00 && netoMesecnaDavcnaOsnova <= 4000.00
izracunDohodnine_05	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova > 4000.00 && netoMesecnaDavcnaOsnova <= 5908.93
izracunDohodnine_06	TRUE	mesecniBrutoDohodek > 0	netoMesecnaDavcnaOsnova > 5908.93

ACTION	ACTION	PRIORITY
$izracunPlace.setZnesekDohodnine($param);	update($param);	
Izračun dohodnine	Nastavitev mesečne olajšave	
new BigDecimal(0.00)	$izracunPlace	40
$izracunPlace.getNetoMesecnaDavcnaOsnova().multiply(new BigDecimal(0.16))	$izracunPlace	40
new BigDecimal(106.95).add(($izracunPlace.getNetoMesecnaDavcnaOsnova().subtract(new BigDecimal(668.44))).multiply(new BigDecimal(0.27)))	$izracunPlace	40
new BigDecimal(385.47).add(($izracunPlace.getNetoMesecnaDavcnaOsnova().subtract(new BigDecimal(1700.00))).multiply(new BigDecimal(0.34)))	$izracunPlace	40
new BigDecimal(1167.47).add(($izracunPlace.getNetoMesecnaDavcnaOsnova().subtract(new BigDecimal(4000.00))).multiply(new BigDecimal(0.39)))	$izracunPlace	40
new BigDecimal(1911.95).add(($izracunPlace.getNetoMesecnaDavcnaOsnova().subtract(new BigDecimal(5908.93))).multiply(new BigDecimal(0.50)))	$izracunPlace	40

Slika 4.14: Pravilo za izračun dohodnine

Nato se sproži pravilo `neto_01` za izračun neto plače iz bruto plače, katerega vidimo na sliki 4.15. Izračuna se po formuli:

`mesecniBrutodohodek-(prispevekDelojemalca+znesekDohodnine)`.

RuleTable izracun_neto	Izračun neto plače		
NAME	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		
	$param	$param	$param
	Izračun plače obstaja	Bruto vrednost plače obstaja	Izračun plače obstaja
neto_01	TRUE	mesecniBrutoDohodek > 0	prispevek != null

ACTION	PRIORITY	NO-LOOP
$izracunPlace.setMesecniNetoDohodek($param);		
Izračun neto plače		
$izracunPlace.getMesecniBrutoDohodek().subtract($izracunPlace.getPrispevek().getZnesekZaDelojemalca().add($izracunPlace.getZnesekDohodnine()))	30	TRUE

Slika 4.15: Pravilo za izračun neto plače

Nato se še sproži pravilo `brutoStrosek_01` za izračun bruto stroška delodajalca, vidno na sliki 4.16. Za izračun bruto stroška delodajalca uporabi formulo:

`mesečniBrutodohodek+prispevekDelodajalca`

RuleTable brutoStrosek	Izračun bruto stroška za delodajalca		
NAME	CONDITION	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()		$prispevek : PrispevekTip from $izracunPlace.prispevek
	$param	$param	$param
	Izračun plače obstaja	Bruto vrednost plače obstaja	Prispevek
brutoStrosek_01	TRUE	mesecniBrutoDohodek > 0	TRUE

ACTION	PRIORITY	NO-LOOP
$izracunPlace.setBrutoStrosekDelodajalca($param);		
Izračun bruto stroška		
$izracunPlace.getMesecniBrutoDohodek().add($prispevek.getZnesekZaDelodajalca())	30	TRUE

Slika 4.16: Pravilo za izračun bruto stroška delodajala

4.4 Sprememba pravil

V primeru, da pride do spremebe v zakonu za obračun plač oziroma višine olajšav, lahko pravila preprosto uredimo v Excel preglednici na sledeč način. Zamislimo si, da je prišel nov zakon za spremembo višine stopenj za splošno olajšavo. In sicer želimo spremeniti višine pogojev za `mesecniBrutoDohodek` v pravilih `splosnaOlajsava_01, splosnaOlajsava_02,` in `splosnaOlajsava_03`. Trenutne vrednosti primerjane v pogoju vidimo v stolpcu `Bruto vrednost plače obstaja` na sliki 4.17. Želimo jih spremeniti na nove vrednosti, zato jih preprosto uredimo v Excel programu in sicer vsako vrstico

popravimo. Popravljeno različico vidimo na sliki 4.18. Nato shranimo, na novo zgradimo Docker vsebnik s KIE izvajalnim strežnikom ter ga postavimo in naša popravljena pravila so nared za izvrševanje. Na ta način lahko poljubno (oziroma v primeru spremembe v zakonu) spremenimo katerokoli pravilo iz nabora naših pravil za izračun plače hitro in preprosto. Tako smo lahko vedno na tekočem in vedno bo naša aplikacija v koraku s časom s trenutnimi zakoni.

RuleTable splosna_davcna_olajsava	Nastavitev splošne olajšave glede na bruto plačo	
NAME	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()	
	$param	$param
	Izračun plače obstaja	**Bruto vrednost plače obstaja**
splosnaOlajsava_01	TRUE	mesecniBrutoDohodek < 930.53
splosnaOlajsava_02	TRUE	mesecniBrutoDohodek >= 930.54 && mesecniBrutoDohodek < 1047.57
splosnaOlajsava_03	TRUE	mesecniBrutoDohodek >= 1047.58

CONDITION	CONDITION
not OlajsavaTip() from $izracunPlace.olajsave	$izracunTip : IzracunTip from $izracunPlace.izracunTip
$param	isIndBrutoPlaca()
Splošna olajšava obstaja	**Izbrana je bruto plača**
tipOlajsave == 'splosnaOlajsava'	TRUE
tipOlajsave == 'splosnaOlajsava'	TRUE

Slika 4.17: Prvotna verzija pravila

RuleTable splosna_davcna_olajsava	Nastavitev splošne olajšave glede na bruto plačo	
NAME	CONDITION	CONDITION
	$izracunPlace : IzracunPlace()	
	$param	$param
	Izračun plače obstaja	**Bruto vrednost plače obstaja**
splosnaOlajsava_01	TRUE	mesecniBrutoDohodek < 975.53
splosnaOlajsava_02	TRUE	mesecniBrutoDohodek >= 975.53 && mesecniBrutoDohodek < 1147.57
splosnaOlajsava_03	TRUE	mesecniBrutoDohodek >= 1147.58

CONDITION	CONDITION
not OlajsavaTip() from $izracunPlace.olajsave	$izracunTip : IzracunTip from $izracunPlace.izracunTip
$param	isIndBrutoPlaca()
Splošna olajšava obstaja	**Izbrana je bruto plača**
tipOlajsave == 'splosnaOlajsava'	TRUE
tipOlajsave == 'splosnaOlajsava'	TRUE
tipOlajsave == 'splosnaOlajsava'	TRUE

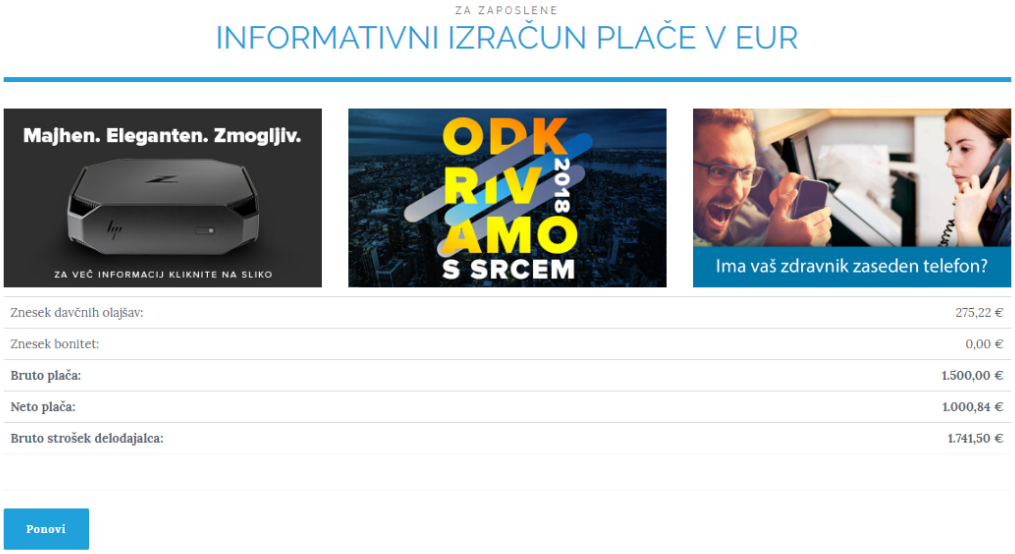
Slika 4.18: Spremenjena verzija pravila

Poglavje 5

Analiza

Našo spletno aplikacijo za izračun plač, ki je izvedba s poslovnimi pravili, bomo primerjali s spletno aplikacijo OPLA podjetja SRC d.o.o, ki jo lahko vidimo na slikah 5.1 in 5.2. Za razliko od naše je aplikacija OPLA v celoti napisana v programskem jeziku Javascript. To pomeni, da je s stališča vzdrževanja in omogočanja sprememb aplikacija, ki uporablja platformo Drools že v osnovi boljša, saj za spremembo pravil preprosto popravimo pravila v Excel datoteki oziroma to stori poslovni analitik.

Slika 5.1: Čelni del spletne aplikacije OPLA



Slika 5.2: Čelni del spletne aplikacije OPLA

5.1 Dobre strani

V primeru spremembe pri aplikaciji OPLA mora poslovni analitik predstaviti spremembe razvijalcu, slednji pa pobrskati po izvorni kodi aplikacije in pravilno umestiti oziroma popraviti pogojni stavek. V tem primeru je večja možnost, da pride do napake, saj se lahko hitro zgodi, da pride do šuma v komunikaciji med razvijalcem in poslovnim analitikom in je potem potrebno večkrat popravljati, kar je lahko zelo zamudno. Tudi izvorna koda naše spletne aplikacije je tudi do 95% manjša, saj vso poslovno logiko in pogojne stavke nadomesti klic na izvajalni strežnik platforme Drools. Primer klica vidimo na sliki 5.3.

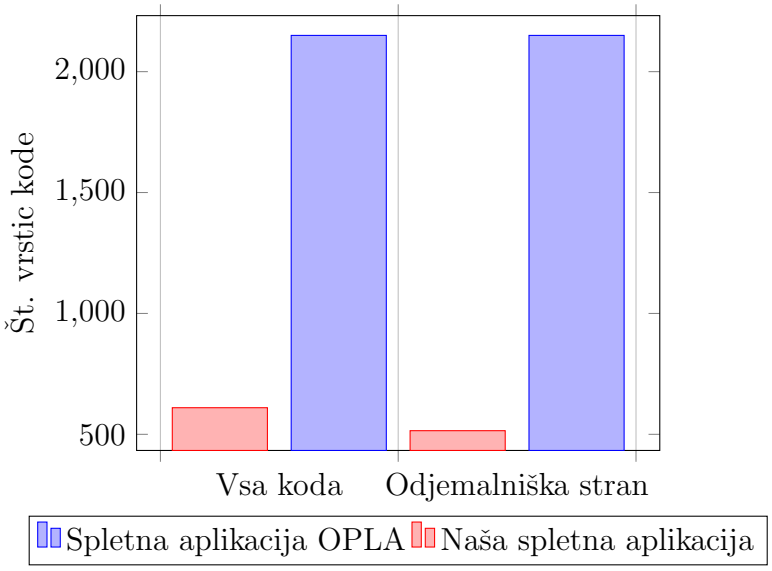
```
$url = "http://10.204.1.242:8080/kie-server-6.5.0.Final-ee7/services/rest/server/containers/instances/opla";
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_HEADER, false);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_HTTPHEADER,
        array("Content-type: application/json","X-KIE-ContentType: JSON","Authorization: Basic [redacted]"));
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($podatki));

$json_response = curl_exec($curl);
$status = curl_getinfo($curl, CURLINFO_HTTP_CODE);
```

Slika 5.3: Klic KIE izvedbenega strežnika

5.1.1 Primerjava po številu vrstic kode

Spletna aplikacija OPLA podjetja SRC vsebuje kar 2150 vrstic kode napisane v programskem jeziku Javascript. Naša spletna aplikacija ima le 95 vrstic kode v programskem jeziku PHP, ker je vsa logika odločanja prestavljena v Excel datoteko.

Ne glede na vse je naša spletna aplikacija dosti preprostejša in bistveno manj zahtevna na strani odjemalca. Če pa v obzir vzamemo še poslovna pravila (napisana v Excelu) in ta za potrebe primerjave pretvorimo v `.drl` datoteko, dobimo `.drl` datoteko s 515 vrstic kode. Skupaj naša aplikacija tako obsega 610 vrstic kode, kar je še vedno 71% manj kot spletna aplikacija podjetja SRC. Primerjavo vidimo na spodnjem grafu 5.4.



Slika 5.4: Primerjava števila vrstic programske kode obeh aplikacij

5.1.2 Integracija s drugimi sistemi

Dobra stran platforme Drools je tudi integracija s drugimi sistemi. Če bi potrebovali ista pravila v neki drugi spletni aplikaciji, bi preprosto uporabili klic do strežnika in dobili izračune. V primeru aplikacije OPLA pa bi bilo potrebno na novo spisati aplikacijo in uporabiti delčke izvorne kode obstoječe aplikacije, kar je bolj zahtevno, in predvsem omogoča veliko napak.

5.1.3 Prilagodljivost platforme Drools

Če želimo ohraniti princip ločevanja poslovne logike iz poslovne aplikacije in ne želimo biti omejeni s strani spletnega vmesnika Drools Workbench, uporabimo enak način, kot smo ga uporabili mi; uporabimo samo izvajalni strežnik s pravili, ki so lahko napisana v jeziku DRL in shranjena v datoteki s končnico `.drl` ali pa v Excel datoteki. Lahko pa preprosto prevedemo Excel datoteko v `.drl` in potem tudi razvojniki nimajo težav s urejanjem pravil, kar se nam zdi prednost uporabe platforme Drools. S tem razbremenimo razvojnike in prihranimo njihov dragocen čas za bolj zahtevne probleme.

5.1.4 Časovna primerjava spremembe poslovnih pravil

Primerjali smo tudi čas porabljen za spremembo poslovnih pravil. Vse povprečne čase smo pridobili s anketo opravljeno v podjetju SRC, ki bo prestavljena v poglavju 5.3. Ko poslovni analitik dobi novo poslovno pravilo (oziroma spremembo v zakonu), ga preuči in pripravi predlog spremembe. Odvisno od obsežnosti novega poslovnega pravila (v našem primeru je preprosto pravilo) mu to vzame v povprečju 35 minut. Ta del priprave predloga spremembe je enak pri obeh aplikacijah.

Aplikacija podjetja SRC - OPLA

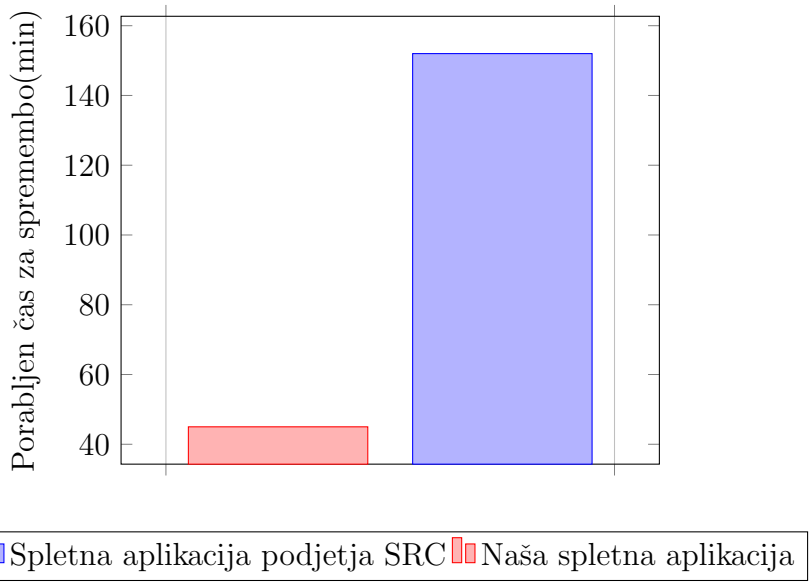
Pri primeru aplikacije OPLA analitik pošlje navodila programerju, katera poslovna pravila in kako jih mora urediti. To vzame analitiku v povprečju 12 minut. Nato je odvisno od odzivnosti in obremenjenosti razvojnika, ampak okviren odzivni čas razvojnikov je 1 uro, v primeru pomembnejših projektov pa se ta čas lahko podaljša tudi do 24 ur. Nato razvojnik s pomočjo navodil popravi potrebna pravila. To v povprečju za naš primer traja 45 min. Ker je naš testni primer bil dokaj preprost, je programer uspešno popravil poslovno pravilo brez dodatne pomoči analitika. Vse skupaj je trajalo 2 uri in 32 minut.

Naša aplikacija

Pri primeru naše aplikacije analitik le odpre Excel datoteko s poslovnimi pravili in poišče poslovno pravilo, za katerega je prej pripravil predlog spremembe. To vzame analitiku v povprečju 10 minut. Ko je našel pravilo, popravi pogoj in potem naloži nazaj pravila na KIE Izvedbeni strežnik. To mu vzame v povprečju 35 minut. Torej celoten postopek spremembe poslovnih pravil za naš primer mu vzame v povprečju 45 minut.

Rezultat primerjave

V primerjavi obeh časov za spremembe poslovnih pravil smo ugotovili, da sprememba v primeru aplikacije, za platformo Drools v povprečju hitrejša za 1 uro in 47 minut. V primeru zasedenosti razvojnikov bi bila ta razlika še večja, vendar je to odvisno od prioritet projektov v podjetju. Primerjavo časov porabljenega za spremembo pravil pri obeh aplikacijah vidimo na sliki 5.5.



Slika 5.5: Primerjava porabljenega časa za spremembo poslovnih pravil

5.2 Slabe strani

Kot vsaka aplikacija ima tudi naša aplikacija slabe strani, ki so se pokazale ob koncu izdelave. Pokazalo se je, da je treba za razvoj Drools aplikacije potrebno imeti razvojnika, ki se vsaj malo spozna na sisteme BRMS ali pa se je pripravljen temu priučiti. Platforma Drools namreč vsebuje določene specifične rešitve, ki jih običajni razvojniki ne poznajo.

5.2.1 Dodatna strojna oprema

Slaba stran naše spletne aplikacije je tudi to, da potrebuje dodatni strežnik, na katerem teče KIE izvajalni strežnik. Ta izvaja poslovna pravila in logiko kot je bilo opisano. Postavitev strežnika lahko podjetje stane nekaj časa in denarja.

5.2.2 Omejenost

V primeru uporabe celostne platforme Drools s spletnim vmesnikom Drools Workbench za vnašanje pravil in pripravo podatkovnih objektov, postane platforma Drools dokaj omejena. Vseeno je primeren za uporabo s strani poslovnih analitikov in ostalih ljudi brez računalniškega predznanja, saj omogoča preprost vnos pravil in podatkovnih objektov. Brez uporabe spletnega vmesnika Drools Workbench za vnašanje poslovnih pravil in podatkovnih objektov je potrebno vsaj osnovno predznanje programskega jezika Java, da lahko ustvarimo bolj zahtevna pravila. To pomeni, da bi tudi poslovni analitiki v tem primeru potrebovali predznanje. Po anketi v podjetju ga večina poslovnih analitikov že ima oziroma so se ga priučili med uporabo platforme Drools.

5.3 Anketa v podjetju SRC

V podjetju SRC smo tudi naredili kratko anketo s delavci na področju poslovnih pravil: izprašali smo 5 analitikov in 5 razvojnikov. Želeli smo izvedeti njihovo mnenje o obeh aplikacijah in platformi Drools.

5.3.1 Primerjava uporabniške izkušnje

Sprva smo jih prosili, naj obe spletni aplikaciji ocenijo vizualno oziroma podajo njihovo uporabniško izkušnjo s obema aplikacijama. Od vseh smo dobili zelo podoben oziroma skoraj enak odgovor. Aplikaciji sta vizualno primerljivi in v koraku s časom, zato nimajo pripomb glede tega.

Ker je naša aplikacija še v razvojni fazi, omogoča le izračun za redno zaposlene. Spletna aplikacija OPLA pa omogoča tudi izračun za študente in za ostale oblike dela (podjemna pogodba, avtorska pogodba, občasno delo). Zato smo jih prosili, da se osredotočijo na izračun za redno zaposlene in podajo svoje mnenje o delovanju obeh spletnih aplikacij. Po podrobnem testiranju (robni primeri, različne višine plač, različne kombinacije bonitet in

olajšav) tako s strani analitikov kot razvijalcev smo dobili odgovore, da obe aplikaciji izračunata natanko enake rezultate. Torej trenutno sta aplikaciji primerljivi oziroma enaki v delovanju in videzu. Tudi časovna odzivnost in hitrost obeh aplikacij je enaka.

5.3.2 Sprememba poslovnih pravil

V naslednjem koraku smo jih prosili, da spremenijo nekaj pravil. Opisali so postopek urejanja pravil v njihovi aplikaciji OPLA.

Postopek je sledeč: najprej poslovni analitiki preučijo nove zakone in pripravijo kot nekakšno psevdo kodo poslovnih pravil oziroma pogojev, na katere vpliva novi zakon. Analitiki morajo biti zato natančni pri opisih, saj je sledeč korak razvojniku pojasniti, kateri pogoji so novi in na kakšen način jih mora spremeniti. Nato je naloga razvojnika, da v programski kodi aplikacije, poišče del oziroma pogojni stavek, ki ga je treba spremeniti. Tukaj so razvijalci še dodali, da zna biti to delo zamudno, predvsem zato, ker lahko pride do napačnega razumevanja vsebine s strani razvijalcev in posledično popravek napačnega pogoja. To pomeni, da se postopek ponovi iz začetka, da se razjasnijo nejasnosti.

Postopek popravljanja poslovnih pravil v primeru naše aplikacije je dosti bolj transparenten. V primeru, ko gre za manjšo spremembo pogoja, poslovni analitik preuči zakon in si pripravi novi pogojni stavek, odpre Excel datoteko s poslovnimi pravili, poišče katerih pravil se novi zakon tiče, jih preprosto spremeni in naloži nazaj na KIE Izvajalni strežnik. V primeru zahtevnejših novih zakonov, je potrebno novo pravilo spisati drugače. Poslovni analitiki lahko pišejo sami, vendar če se jim kje zatakne za pomoč poprosijo razvojnike. Vloga razvojnika je tu bolj pomoč kot pa implementacija oziroma popravljanje poslovnih pravil. Sodelovanje razvojnikov je zelo pomembno, saj v primeru popravljanja pravil s strani analitika ta pogosto pusti napako v pravilih. Tu vstopi razvojnik, ki z naprednim znanjem razhroščevanja hitreje odkrije napako od poslovnega analitika.

Nato smo vprašali razvojnike, kdo naj implementira popravke v program-

skem jeziku Javascript: sami ali pa bi raje bili kot pomoč poslovnim analitikom pri zahtevnejših pravilih. Od vseh razvijalcev sem dobil enak odgovor. Raje bi bili kot pomoč poslovnim analitikom, saj imajo dosti še drugih projektov na skrbi in bi jih to zelo razbremenilo. Poslovni analitki so se pa v veliki meri strinjali, da bi tudi njim bilo lažje, da sami uredijo pravila, saj oni bolj razumejo vsebinsko stališče pogojev in se ni treba truditi, na kakšen način bodo to razložili razvijalcem.

5.3.3 Zaključek ankete

Ob koncu ankete je bilo jasno, da bi razvojniki in poslovni analitiki raje izbrali spletno aplikacijo, ki uporablja platformo Drools, saj je s tem omogočeno lažje vzdrževanje in prilagajanje poslovnih pravil. Razvijalcem smo omenili tudi slabosti naše spletne aplikacije. Vseeno so menili, da je smiselno uporabiti večino kode na strani strežnika, saj v primeru uporabnikov, ki dostopajo s starejših računalnikov in slabše omrežne povezave, lahko pride do zamud pri nalaganju spletne aplikacije. Ta namreč temelji na velikem številu vrstic kode napisane v programskem jeziku Javascript, ki se naloži ob obisku spletne aplikacije.

Če povzamemo rezultate ankete, lahko ugotovimo, da bi s pomočjo platforme Drools, hitreje in učinkoviteje naredili spletno aplikacijo za izračun plač. S tem bi razbremenili razvijalce, poslovni analitiki pa bi imeli večjo kontrolo nad poslovnimi pravili.

Poglavje 6

Zaključek

Ustvarili smo delujoč primer kalkulatorja plač s pomočjo platforme Drools, ki lahko izračuna znesek neto plače iz bruto plače in obratno, hkrati pa upošteva olajšave kot so otroci, invalidnost in druge. Zaradi uporabe poslovnih pravil in platforme Drools je vzdrževanje spletne aplikacije in poslovnih pravil preprosto. Naša aplikacija ima kar 71% odstotkov krajšo izvorno kodo kot izvorna koda primerljive spletne aplikacije napisane v programskem jeziku Javascript. S pomočjo primerjave spremembe pravil smo ugotovili, da podjetje lahko prihrani kar 70% časa za spremembo pravil, če to opravijo poslovni analitiki s pomočjo platforme Drools. Izjema je le to, da je potrebno postaviti dodaten izvedbeni strežnik s poslovnimi pravil.

Potrebno je tudi imeti razvojnika s znanjem o platformi Drools, vendar smo po podrobni ugotovili, da se lahko hitro priučimo uporabe platforme Drools in njenih orodij. V primeru novega zakona za izračun plače ali pa novega zakona o višini olajšav preprosto odpremo `.drl` ali Excel datoteko s pravili, poiščemo pravilo, ki vsebuje podatke, ki se bodo spremenili, jih spremenimo in osvežimo pravila na izvajalnem strežniku. Pravila lahko tudi urejajo poslovni analitiki preko programa Excel, ki omogoča pregleden pregled vseh pravil.

Z anketo med zaposlenimi v podjetju SRC smo ugotovili, da bi uporaba plaforme Drools omogočila lažje vzdrževanje spletne aplikacije in hitrejšo ter

preprostejšo urejevanje pravil. Spoznali smo tudi, da bi bili razvijalci bolj razbremenjeni, saj skrbijo za več različnih projektov in če bi poslovni analitiki skrbeli za poslovna pravila. Tudi razvijalci bi bili bolj razbremenjeni in bi s tem prihranili na času, ki bi ga porabili za ostale projekte. Razvijalci so tudi potrdili, da uporaba dodatnega izvajalnega strežnika ne bi bila problematična, ampak celo priporočljiva.

Lahko bi še za primerjavo poizkusili vnesti vsa pravila preko spletnega vmesnika Drools Workbench, ampak po anketi v podjetju SRC, kjer uporabljajo platformo Drools in BRMS sisteme že nekaj časa, zato smo se raje odločili za opcijo,ki omogoča hitrejšo in lažjo pripravo tudi zelo zahtevnih pravil.

V praksi se BRMS sistemi uporabljajo že dlje časa predvsem za bančne in finančne projekte, kjer se pravila nenehno spreminjajo. Mislimo, da je uporabno, da se vpelje uporabo platforme Drools tudi v druge projekte, kjer je njihova uporaba smiselna (predvsem, da imajo veliko pogojev, ki se lahko spremenijo v nekem določenem časovnem obdobju). Žal se pa pojavi težava, če so poslovni analitiki podhranjeni z znanjem o kakšnem programskem jeziku in jih je potrebno izobraziti na področju znanja programskih jezikov, seveda v primeru uporabe zahtevnejših poslovnih pravil napisanih v programskem jeziku DRL. Platforma Drools je vsestransko uporabna, najbolje pa se izkaže v primeru stalnih sprememb v poslovnih pravilih, kjer omogoči lažje vzdrževanje in spreminjanje le-teh.

V okviru diplomske naloge bi lahko podrobneje raziskali še druge dele platforme Drools, kot je recimo Drools Workbench. Ta bi lahko še primerjali kako zahtevna pravila je možno vnašati preko spletne aplikacije za upravljanje s poslovnimi pravili. Nato bi primerjali delovanje vseh treh sistemov in s tem bi ugotovili, kateri način je najbolj smiseln za vpeljavo uporabe sistema BRMS. Za primerjavo bi lahko tudi uporabili še katerega od plačljivih sistemov BRMS, vendar so žal ti sistemi zelo dragi in si tega ne moremo privoščiti. Predvsem so namenjeni točno določenim problematikam in niso tako razširljivi in vsestranski za uporabo kot platforma Drools. Pokazali bi

lahko tudi še primer razhroščevanja poslovnih pravil s pomočjo integracije s razvojnim okoljem Eclipse, ki zelo prav pride pri napakah na zahtevnejših poslovnih pravilih. V primeru dobrega poslovnega procesa bi lahko prikazali tudi integracijo s orodjem jBPM, ki je del platforme Drools, saj omogoča avtomatiziranje poslovnih procesov.

Literatura

[1] Apache docker image. Dosegljivo: https://hub.docker.com/_/httpd. [Dostopano 20. 12. 2018].

[2] Bootstrap. Dosegljivo: https://getbootstrap.com/. [Dostopano 20. 12. 2018].

[3] BUSINESS RULE MANAGEMENT SYSTEM. Dosegljivo: https://www.hartmannsoftware.com/Blog/Enterprise-Rule-Applications/brms. [Dostopano 20. 12. 2018].

[4] Business rules approach. Dosegljivo: https://en.wikipedia.org/wiki/Business_rules_approach. [Dostopano 20. 12. 2018].

[5] Business rules group. Dosegljivo: http://www.businessrulesgroup.org/theBRG.htm. [Dostopano 20. 12. 2018].

[6] Business rules manifesto. Dosegljivo: http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf. [Dostopano 20. 12. 2018].

[7] DAVKI NA 1, 2, 3 - ZA FIZIČNE OSEBE. Dosegljivo: http://www.fu.gov.si/fileadmin/Internet/Uvodne_strani/Prebivalci/davki123.pdf#page=7. [Dostopano 20. 12. 2018].

[8] Davčne olajšave invalidov. Dosegljivo: http://www.zvd.si/media/medialibrary/2018/01/Zavod_za_varstvo_pri_delu_RDV_3_2010_Davcne_olajsave_invalidov.pdf. [Dostopano 20. 12. 2018].

[9] Docker Documentation. Dosegljivo: https://docs.docker.com/. [Dostopano 20. 12. 2018].

[10] Drools. Dosegljivo: https://www.drools.org. [Dostopano 20. 12. 2018].

[11] Drools - Rules Writing. Dosegljivo: https://www.tutorialspoint.com/drools/drools_rules_writing.htm. [Dostopano 20. 12. 2018].

[12] Drools - The rule language. Dosegljivo: https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch05.html. [Dostopano 20. 12. 2018].

[13] Drools Documentation. Dosegljivo: https://docs.jboss.org/drools/release/6.5.0.Final/drools-docs/html_single/. [Dostopano 10. 11. 2018].

[14] Drools Expert. Dosegljivo: https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/. [Dostopano 20. 12. 2018].

[15] FICO Blaze Advisor Decision Rules Management System. Dosegljivo: https://www.fico.com/en/products/fico-blaze-advisor-decision-rules-management-system. [Dostopano 20. 12. 2018].

[16] HTML5 Tutorial. Dosegljivo: https://www.w3schools.com/html/. [Dostopano 20. 12. 2018].

[17] IBM Operational Decision Manager. Dosegljivo: https://www.ibm.com/us-en/marketplace/operational-decision-manager. [Dostopano 20. 12. 2018].

[18] InRule. Dosegljivo: https://www.inrule.com/. [Dostopano 20. 12. 2018].

[19] JESS. Dosegljivo: https://www.jessrules.com/jess/index.shtml/. [Dostopano 20. 12. 2018].

[20] KIE Execution Server. Dosegljivo: https://docs.jboss.org/drools/release/6.2.0.Final/drools-docs/html/ch19.html. [Dostopano 20. 12. 2018].

[21] Microsoft Excel. Dosegljivo: https://products.office.com/sl-si/excel. [Dostopano 20. 12. 2018].

[22] OpenRules. Dosegljivo: http://openrules.com/. [Dostopano 20. 12. 2018].

[23] PHP. Dosegljivo: http://www.php.net/. [Dostopano 20. 12. 2018].

[24] Pravilnik o določitvi olajšav in lestvice za odmero dohodnine. Dosegljivo: http://www.pisrs.si/Pis.web/pregledPredpisa?id=PRAV12719. [Dostopano 20. 12. 2018].

[25] Progress Corticon. Dosegljivo: https://www.progress.com/corticon. [Dostopano 20. 12. 2018].

[26] Rete Algorithm. Dosegljivo: https://access.redhat.com/documentation/en-us/red_hat_jboss_bpm_suite/6.2/html/development_guide/sect-rete_algorithm. [Dostopano 20. 12. 2018].

[27] What is a Business Rules Management System (BRMS)? Dosegljivo: https://www.progress.com/faqs/corticon-faqs/what-is-a-business-rules-management-system. [Dostopano 10. 11. 2018].

[28] WildFly. Dosegljivo: http://www.wildfly.org/. [Dostopano 20. 12. 2018].

[29] Wildfly docker image. Dosegljivo: https://hub.docker.com/r/jboss/wildfly/. [Dostopano 20. 12. 2018].

[30] Paul Browne. *JBoss Drools Business Rules*. 2009.

[31] Sean P. Kane KarlMatthias. *Docker: Up & Running: Shipping Reliable Containers in Production 1st Edition*. 2015.

[32] Mariano De Maio Mauricio Salatino, Esteban Aliverti. *Mastering JBoss Drools 6*. 2016.

[33] Vidhita Deshmukh. BRMS (Business Rule Management System). Dosegljivo: `http://techxpla.com/2017/10/25/business-rule-management-system/`. [Dostopano 10. 11. 2018].